Gonçalo Moreira Cândido

Licenciado em Engenharia Electrotécnica e de Computadores

Service-oriented Architecture for Device Lifecycle Support in Industrial Automation

Dissertação para obtenção do Grau de Doutor em
Engenharia Electrotécnica e de Computadores
Especialidade: Robótica e Manufactura Integrada

Orientador : José Barata de Oliveira,
Professor Doutor, FCT-UNL

Júri:

Presidente:	Prof. Doutor Luís Manuel Camarinha-Matos
Arguentes:	Dr. Christoph Hanisch
	Prof. Doutor Carlos Eduardo Pereira
Vogais:	Prof. Doutor José António Barata de Oliveira
	Prof. Doutor Paulo Jorge Pinto Leitão
	Dr.-Ing Armando Walter Colombo
	Doutor Luís Domingos Ferreira Ribeiro

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Dezembro de 2013

Gonçalo Moreira Cândido

Licenciado em Engenharia Electrotécnica e de Computadores

Service-oriented Architecture for Device Lifecycle Support in Industrial Automation

Dissertação para obtenção do Grau de Doutor em
Engenharia Electrotécnica e de Computadores
Especialidade: Robótica e Manufactura Integrada

Orientador : José Barata de Oliveira,
Professor Doutor, FCT-UNL

Júri:

Presidente:	Prof. Doutor Luís Manuel Camarinha-Matos
Arguentes:	Dr. Christoph Hanisch
	Prof. Doutor Carlos Eduardo Pereira
Vogais:	Prof. Doutor José António Barata de Oliveira
	Prof. Doutor Paulo Jorge Pinto Leitão
	Dr.-Ing Armando Walter Colombo
	Doutor Luís Domingos Ferreira Ribeiro

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Dezembro de 2013

Service-oriented Architecture
for Device Lifecycle Support in Industrial Automation

Aos meus pais

Acknowledgements

Since that day in 2005 when I decided along with my dear friend Filipe Feijão that we should put our hands to work and try to reactivate the Novaflex shop floor, Prof. José Barata has been the only academic supervisor I had the incredible privilege to work with. On that particular day I was very far from imagining the long and rewarding path that would get me to this moment, so that is why first and foremost I must express my gratitude and appreciation to my dear supervisor for all the unique opportunities and moments of friendship and scientific endeavours along these last years.

Alongside with Prof. José Barata guidance, I must also recognise the contribution of François Jammes and Prof. Walter Colombo for their support during my extraordinary experience at Schneider Electric DInnov department in Grenoble, France. While working integrated in the "Web Services Project" team, the remarkable constructive criticism, vision and availability by François Jammes enriched my days living abroad for both service-oriented discussions or vital snow and mountain advice. Even with a border separating France from Germany, Prof. Walter Colombo managed to contaminate everybody with his enthusiasm and challenges to pursue innovative solutions by thinking "outside of the box". In the context of my days at *DInnov*, I also want to express my gratitude for all the help and good humour from my closest colleagues Fabrice Roudet, Dimitri Aujol, Sylvain Meygret, Jaime Sanchez-Laulhe, André Guérard, Laurent Marsala, Fabrice Depeisses, Gérome Hamel, Philippe Nappey, Patrick Allard-Jacquin and Bernard Bony.

I would like to personally thank my "senior" colleagues at Uninova, Pedro Santana e Luís Ribeiro, for their friendship and passionate discussions on both scientific or any other social-cultural theme. During this period I have shared the working environment with other amazing people that really allowed me to work in a constructive environment. For this and much more I would like to thank Pedro Deusdado, Pedro Gomes, Eduardo Pinto, Magno Guedes, Nelson Alves, Ricardo Mendonça, Francisco Marques and André Lourenço. A special word of appreciation to my dear colleagues and friends Giovanni Di Orio and Carlos Filipe Sousa for all the shared work in the scope of Self-Learning project and validation of part of this thesis results, respectively.

In the months that preceded the final presentation of this work, Rita Cabral Barata

had a crucial role on its final success. Her continuous loving support and encouragement were essential for me to continue motivated and sure of a positive conclusion.

Outside of the university sphere I always had my beloved family supporting and encouraging me in every decision I took along the years but never without offering their opinion and advice on how to surpass occasional trepidations along the way. For this and for all the rest along these 30 years, I want to foremost thank my parents Albertina Moreira and Jaime Cândido for empowering me to be who I am today.

Obrigado.

"Divide each difficulty into as many parts as is feasible and necessary to resolve it. "

René Descartes
in Discourse on the Method

Abstract

This thesis addresses the device lifecycle support thematic in the scope of service-oriented industrial automation domain. This domain is known for its plethora of heterogeneous equipment encompassing distinct functions, form factors, network interfaces, or I/O specifications supported by dissimilar software and hardware platforms. There is then an evident and crescent need to take every device into account and improve the agility performance during setup, control, management, monitoring and diagnosis phases.

Service-oriented Architecture (SOA) paradigm is currently a widely endorsed approach for both business and enterprise systems integration. SOA concepts and technology are continuously spreading along the layers of the enterprise organization envisioning a unified interoperability solution. SOA promotes discoverability, loose coupling, abstraction, autonomy and composition of services relying on open web standards – features that can provide an important contribution to the industrial automation domain.

The present work seized industrial automation device level requirements, constraints and needs to determine how and where can SOA be employed to solve some of the existent difficulties. Supported by these outcomes, a reference architecture shaped by distributed, adaptive and composable modules is proposed. This architecture will assist and ease the role of systems integrators during reengineering-related interventions throughout system lifecycle. In a converging direction, the present work also proposes a service-oriented device model to support previous architecture vision and goals by including embedded added-value in terms of service-oriented peer-to-peer discovery and identification, configuration, management, as well as agile customization of device resources.

In this context, the implementation and validation work proved not simply the feasibility and fitness of the proposed solution to two distinct test-benches but also its relevance to the expanding domain of SOA applications to support device lifecycle in the industrial automation domain.

Keywords: Device lifecycle support, Service-oriented Architecture, device model, reference architecture, industrial automation.

Resumo

Esta tese aborda a gestão de dispositivos ao longo do seu ciclo de vida no domínio da automação industrial orientada a serviços. Este domínio é reconhecido pela multiplicidade de equipamentos heterogéneos incorporando diversas funcionalidades, factores de forma, interfaces de rede ou especificações de I/O sustentadas por diferentes plataformas de software e hardware. Existe então uma evidente e crescente necessidade de acesso a estes dispositivos de forma a melhorar os níveis de agilidade durante as fases de instalação, controlo, monitorização, diagnóstico e reconfiguração.

O paradigma da Arquitectura orientada a Serviços (SOA - Service-oriented Architecture) é atualmente uma solução amplamente utilizada na integração de sistemas ao nível comercial e empresarial. Os conceitos e tecnologia SOA continuam a ser difundidos pelos diferentes níveis da estrutura empresarial ambicionando uma solução de interoperabilidade unificada. Os príncipios SOA promovem a descoberta, desacoplamento, abstração, autonomia e composição de serviços com base em tecnologias abertas de rede – características estas que poderão oferecer um contributo relevante ao domínio de automação industrial.

O presente trabalho congloba os requisitos, restrições e aspirações do domínio da automação industrial ao nível do dispositivo e determina como e onde a abordagem SOA pode ser empregue para reduzir algumas das dificuldades existentes. Baseado nesta premissa foi proposta uma arquitectura de referência composta por um conjunto de entidades distribuídas combináveis e adaptáveis. Esta arquitectura vai assistir e simplificar o papel de integrador de sistemas durante as diversas intervenções no sistema ao longo do ciclo de vida do mesmo. Numa linha confluente, este trabalho propõem também um modelo orientado a serviços para dispositivos de forma a suportar os objectivos e visão da arquitetura proposta, incorporando uma mais valia em termos de descoberta, identificação, instalação e gestão de dispositivos, bem como na ágil customização dos recursos nestes existentes.

Neste contexto, o trabalho de implementação e validação revelou não apenas a viabilidade e conformidade da solução proposta em duas plataformas de teste distintas, mas também a sua relevância no domínio crescente das aplicações SOA vocacionadas à gestão do ciclo de vida de um dispositivo no âmbito da automação industrial orientada a serviços.

Palavras-chave: Suporte ao ciclo de vida de um dispositivo orientado a serviços, Arquitectura orientada a Serviços, modelo de dispositivo, arquitectura de referência, automação industrial.

Contents

List of Figures

List of Tables

Listings

Glossary of Abbreviations

AI	Artificial Intelligence
AmI	Ambient Intelligence
API	Application Programming Interface
BPEL	Business Process Execution Language
BPML	Business Process Modelling Language
BPMN	Business Process Modelling and Notation
CIM	Computer Integrated Manufacturing
CPU	Central Processing Unit
DLL	Dynamic Link Library
DCS	Distributed Control System
DPWS	Devices Profile for Web Services
EAS	Evolvable Assembly Systems
EIA	Enterprise Information Architectures
EPS	Evolvable Production Systems
ERP	Enterprise Resource Planning
ESB	Enterprise Service Bus
FDR	Faulty Device Replacement
FP6	Sixth Framework Program
FP7	Seventh Framework Program

GUI	Graphical User Interface
HMI	Human-Machine Interface
HTTP	Hypertext Transfer Protocol
ICT	Information and Communication Technology
IDE	Integrated Development Environment
IE	Industrial Ethernet
IP	Internet Protocol
LAN	Local Area Network
MAS	Multi-Agent System
MEP	Message Exchange Pattern
MES	Manufacturing Execution System
OEM	Original Equipment Manufacturer
OOP	Object-Oriented Programming
OPC	Object Linking and Embedding (OLE) for Process Control
OPC UA	OPC Unified Architecture
OS	Operating System
OWL	Web Ontology Language
OWL-S	Semantic Markup for Web Services
PLC	Programmable Logic Controller
QoS	Quality of Service
RDQL	RDF Data Query Language
REST	REpresentational State Transfer
RFID	Radio-Frequency Identification
RPC	Remote Procedure Call
RTU	Remote Terminal Unit
SCA	Service Component Architecture
SCADA	Supervisory Control And Data Acquisition

SME	Small/Medium Enterprise
SNMP	Simple Network Management Protocol
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SOC	Service-Oriented Computing
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UDDI	Universal Description Discovery and Integration
UML	Unified Modelling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WSDL	Web Service Description Language
XML	eXtensible Markup Language
6LoWPAN	IPv6 over Low power Wireless Personal Area Networks

1

Introduction

1.1 Background

The world of embedded computing is characterized by a high degree of diversity in device functionality, form factor, network protocols, input/output features, as well as for the presence of several unlike hardware and software platforms. In areas with a large base of installed devices like industrial automation, this has often resulted in a collection of heterogeneous environments poor in scalability, reuse, interoperability and ability to adapt to new requirements or production variations. In the particular domain of manufacturing, major players are progressively recognizing this situation and try to adapt their equipment and integration solutions offer to help customers to cope with these matters. At organization level, managers already noticed that they need to interact and cooperate with other organizations in order to remain competitive as discussed by [Camarinha-Matos and Afsarmanesh, 2005, Vernadat, 2007, Gunasekaran et al., 2008, Sarimveis et al., 2008]. Nevertheless, at device level, these solutions mostly rely on updates to existing product offers and do not translate into a true shift of Information and Communication Technology (ICT) or production paradigm.

Interoperability tends to be considered simply as a technical issue and its real implications are sometimes underrated. Interoperability, as in [IEEE, 1990], refers to the ability of two or more systems or components to exchange information and to use the information that has been exchanged despite the differences in language, interface and execution platform. A truly interoperable organisation is able to maximise the value and reuse the potential of information under its control. However, interoperability is extremely dependent on content semantics more than over simple software or hardware compatibility [Park and Ram, 2004].

Semantic interoperability emerged as the ability to automatically interpret the exchanged information meaningfully and precisely as expected in involved systems or components [Heiler, 1995, Ouksel and Sheth, 1999]. In this context, the exchanged information must be unambiguously defined, i.e. what is perceived by the transmitter is the same as what is understood by the receiver.

Along with interoperability, the concept of agility implies being more than simply flexible or lean. Flexibility refers to the ability exhibited by a company that is able to adjust itself to produce a predetermined range of solutions or products [Sethi and Sethi, 1990, Gupta and Goyal, 1989], while lean essentially means producing without waste [Shah and Ward, 2003]. On the other hand, agility relates to operating efficiently in a competitive environment dominated by change and uncertainty [Goldman et al., 1995]. An agile company must be able to quickly adapt to face challenges determined by globalization, environmental and working conditions regulations, improved standards for quality, fast technological mutation along with product variations in terms of form, functionality and throughput [Hayes and Pisano, 1994, Lin et al., 2006].

In spite of being programmed to be predictable, history proves that unexpected behaviour arises, equipment fails and changes on requirements, which imply a continuous need to reengineer these systems. Uncertainty must be always taken into account and reengineering processes need to be sufficiently agile to cope with the evolution of production requirements. Furthermore, industrial automation applications are today mostly structured in a rigid pyramidal hierarchical approach, which also compromise the ability to perform lifecycle revisions from top business layers to shop floor device level. When a lower level entity cannot accomplish the desired agility target, the overall system will be incapable of delivering the expected performance, i.e. the agility of a complete system is always constrained by its least agile element.

This way, uncertainty must be always taken into account and reengineering processes need to be sufficiently agile to cope with the evolution of production requirements. It is then important to enhance the agility across and between all company ICT layers, from Enterprise Resource Planning (ERP) to field device level to become a whole agile entity in which its elements interact in a seamless and harmonized manner. This way, the device level in industrial automation plays a fundamental role on supporting overall company agility, since a device is the last frontier where higher level process requirements, guidelines and workflows are transformed into a structured collection of physical actions and procedures. Management, monitoring and control aspects are then fundamental to be taken into account in order to deliver a good level of agility at device level.

Firstly referred in [Schulte and Natis, 1996], Service-Oriented Architecture (SOA) paradigm has emerged and rapidly grown as a standard solution for publishing and accessing information in an increasingly Internet-ubiquitous world. This new approach defines standard interfaces and protocols that allow developers to encapsulate functions and tools as services that clients can access without knowledge or control over their concrete implementation as depicted in [Papazoglou and Georgakopoulos, 2003, Foster,

2005].

SOA establishes an architectural model that aims to enhance the efficiency, interoperability, agility, and productivity of an enterprise by positioning services as the primary means through which solution logic is represented in support of the realization of strategic goals associated with service-oriented computing [Erl, 2005, Erl, 2007].

The increasingly interest in SOA has been stimulated by an influential industry trend: Web Services technology [W3C, 2002]. Although Web Services do not necessarily translate to SOA, and not all SOA is based on Web Services, these last ones recognition are helping to bring SOA to a wider audience, while SOA concepts and principles will contribute to more successful web services deployments [Josuttis, 2007]. As for other domains, service-oriented approaches are now entering the industrial automation domain in a top-down way [Jammes and Smit, 2005b]. This recent approach at device level has a direct impact on how industrial automation applications will be designed and developed. A complete industrial automation application comprises a smooth integration and alignment between complex high level management layers and field level automation behaviour. Since SOA promotes transparency and a wide range of communication forms between different infrastructure elements it would be possible to picture an improved communication mean between the lowest sensor to the highest ICT system. This prospect can offer more sophisticated high level services and support a more reliable cross-layer decision-making while breaking the traditional industrial pyramid to allow the emergence of new fully distributed architectures.

In this context, a device and its hosted services are the core building blocks of a service-oriented industrial automation system. The use of some embedded intelligence alongside the infrastructure and tools that ease the access and management of devices enables more autonomous, interoperable and self-contained devices. This way, the task of managing the system along its lifecycle becomes simplified by only having to handle more abstract and complex information instead of low level intricacies of devices. Also, devices easier to setup, debug, monitor and diagnose are a key-factor during (re)engineering or down-time phases by saving a considerable amount of integration time and cost that significantly reduces productivity rates.

To promote system agility, engineering tools should straightforwardly and intuitively allow the integrator to build the desired application easily and faster when compared to older approaches with, at least, the same level of robustness and performance – this is the key aspect to a wider adoption of SOA in any domain of application [Bloomberg and Schmelzer, 2006].

1.2 Research Problem

Taking into account previous background, it is fundamental to understand the concrete requirements and progress expectations – **how can SOA paradigm be used to address industrial automation requirements and meet productivity potential?**

In the industrial automation domain, one of the major goals is to be able to effortlessly reach, control and manage each device existing on the shop floor along system lifecycle. This factor has a direct impact on the ability of a modern company to cope in an agile way with changeable system requirements and processes, machines maintenance and unexpected down-times while ensuring a competitive and reliable performance.

RQ.1 – What are the key requirements and features still required to be addressed at device level in service-oriented industrial automation context?

There is very little doubt that the SOA principles and technology will continue to have a major impact in many branches of technology, not exclusively in original business ICT domain, but also in other areas where these methodologies can be adapted to. One of the most promising approaches is its application at automation device level where the usage of high level service-oriented communications infrastructure is expected to deliver an important input. Due to automation domain specificity, it is important to research the compliance between these and SOA principles, methodology and technology, adapting them when needed.

H.1 – The result of the retrieval and assessment of the key requirements and features that still need to be addressed at the industrial automation device level will be enriched if both SOA and industrial automation research vectors are combined and inferred which SOA concepts, principles and technology better fit the industrial automation domain.

After specifying the set of requirements and features essential to industrial automation domain, and how to accordingly adapt SOA concepts, principles and technology, there is a need to **define a compliant service-oriented infrastructure to support system lifecycle management and evolution**.

RQ.2 – Which tools and services are essential to support device lifecycle assistance of a service-oriented application in industrial automation?

The methodology and set of guidelines provided as the result of **H.1** will conduct the specification of this set of services and tools. These will constitute an infrastructure to assist the integrator during reengineering tasks of the system device level along its lifecycle. This service-oriented infrastructure will focus on device setup, monitoring and management, while enabling a more effortless and transparent interoperability between all ecosystem entities. Following SOA principles, this infrastructure must be also adjustable to fit particular system specificity.

One of the most crucial features relates with the ability to manage and deploy services into a device in a generic way. It will increase system agility during reengineering interventions, while it remains compliant with open web standards, which also increases device interoperability across different vendors. Besides the need to easily deploy services, there is an emergent need to improve, and even automate, the processes of identification, setup, monitoring and management of these resources in an agile manner.

H.2 – The ability to assist and even automate reengineering interventions along the lifecycle of a service-oriented industrial automation system is improved if a modular and customizable service-oriented reference architecture is available and composed by an interoperable set of tools and services aimed to support the device lifecycle management and evolution.

The reference architecture referred in **H.2** must be deployed on top of a device level infrastructure compliant with SOA concepts and principles. In this context, and taking previous considerations into account, a device is to be seen as the main logical entity that abstracts a system resource, while its services represent the functionalities it offers to the system. A service encapsulates a function or behaviour that can be discovered and employed by another entity during the execution of a particular task.

This way it is fundamental to **research and specify a new device model** that will embed the appropriate set of features and skills to support a harmonious integration in a service-oriented industrial automation environment.

RQ.3 – What model should be employed to better exploit SOA at device level in service-oriented industrial automation?

The proposed device model must lay the foundations upon which the above infrastructure will rely on. Since the majority of business level ICT is already supported by SOA technology, it is important to investigate the device level from industrial automation domain and make it effortlessly interoperable in a cross-layer manner, while providing the expected functionality and performance to domain experts and end-users.

This way, the proposed device model must provide the means, i.e. services, which will allow a systems integrator to manage each device in an agile manner using the tools and services depicted on the reference architecture. By embedding the ability to effortlessly discover and manage each device, and subsequently the overall system, the device "plug-and-play" factor is expected to increase as well as the system agility to cope with lifecycle reengineering interventions.

Furthermore, besides these "out of the box" features available as services hosted in each device, it is fundamental to provide the means for an user to deploy and manage

his own services into a device. Not only services related to the control or monitoring of the current application, but also other services that can improve device performance and features.

The model should remain abstract enough to be mapped to a wider range of application domains and also to legacy equipment.

H.3 – At device level in industrial automation, service-oriented compliancy and "plug-and-play" agility are increased if a service-oriented device model is specified comprising a set of built-in generic services to support agile discovery, identification, setup, monitoring and diagnosis along with the ability to deploy and manage application-specific services.

1.3 Aimed Contributions

The current work intends to clarify the application of a SOA approaches at device level in the industrial automation domain by proposing a reference architecture and a device model aimed to enhance the management and lifecycle of a service-oriented system during its lifecycle. The current work contributed in the following subjects:

- **C.1 – SOA in industrial automation analysis** – After surveying industrial automation domain in terms of requirements, expectations and trends, this work presents a discussion on how SOA aspects can be employed or adapted to address these issues exposing at the same time their major advantages, drawbacks and open aspects.
- **C.2 – Reference Architecture** – the proposed architecture was developed by specifying and assembling elements that can mutually interact and be combined to agile system evolution at industrial automation device level. The reason to define a distributed collection of entities is to allow an adaptive but still customized cast according to system requirements and constraints. The specification of each architecture element is achieved by taking into account the common needs and tasks performed along system lifecycle in the majority of automation systems. The application of this integrated approach will provide a major input over commissioning, setup and reengineering phases in industrial automation devices lifecycle.
  - **C.2.1 – Semantic Assistance** – By relying on semantic tags embedded in each device and hosted service, it would be possible to apply semantic web methodologies to cope with information uncertainty and mismatches. The present architecture introduces elements that, supported by these methodologies, can be combined and interact to provide a major contribution over device discovery and identification phases, as well as for service interface matching between different devices and services providers.

- **C.3 – Device Model** – Exploiting **C.1** conclusions and taking into account the reference architecture prospects of **C.2**, a device model is proposed envisioning an integrated approach to support SOA at device level. The device model is open and adaptive to face system variations along its lifecycle, and even to other domains of application. The definition of a device model compliant at the same time with SOA principles and industrial automation domain requirements can turn out to be the key factor to allow a definitive and solid expansion of the SOA approach into device level.

  - **C.3.1 – Generic Services** – Exposing generic device functions as services augments device interoperability, as it also increases device added value and opens new business opportunities. This set of generic services includes a deployment service that allows the user to deploy and manage its own application services.
  - **C.3.2 – Domain backward compliance** – Since services rely on open web standards, its interfaces and communication practises promote abstraction and independence from concrete implementation. This feature complies with more traditional approaches by allowing the use of domain programming languages, such as IEC61131-3, to develop control procedures and reuse of pre-existing code.
  - **C.3.3 – Merge of DPWS & OPC UA specifications** – At device level in industrial automation, Devices Profile for Web Services (DPWS) and OPC Unified Architecture (OPC UA) specifications are considered the principal solutions to deploy SOA-based applications. The present work includes a deep analysis of both specifications alongside other complimentary WS-* specifications and provides a convergence approach to enable a unified SOA-based approach.

In summary, the present work tackles three major axes: a set of principles and guidelines on how to exploit SOA in industrial automation device level supported by a domain survey, a lifecycle support reference architecture and a compliant service-oriented device model as illustrated in Figure 1.1.

1.4 Domain Impact

Even if the current work is focused on industrial automation device level and adjacent infrastructure, it is arguable that these machines and processes are not decisive forces of production, but simply their reflection as stated in [Noble, 2011]. Technology related developments are always mediated by social power and market domination altogether with the natural contradictions of the advantages and return on investment versus widespread deployed methods and technologies. Having this in mind, the research community, and particularly academia players, must still pursue the efforts of education and dissemination to concretely exploit the corroborated methods and results. This exploitation is the

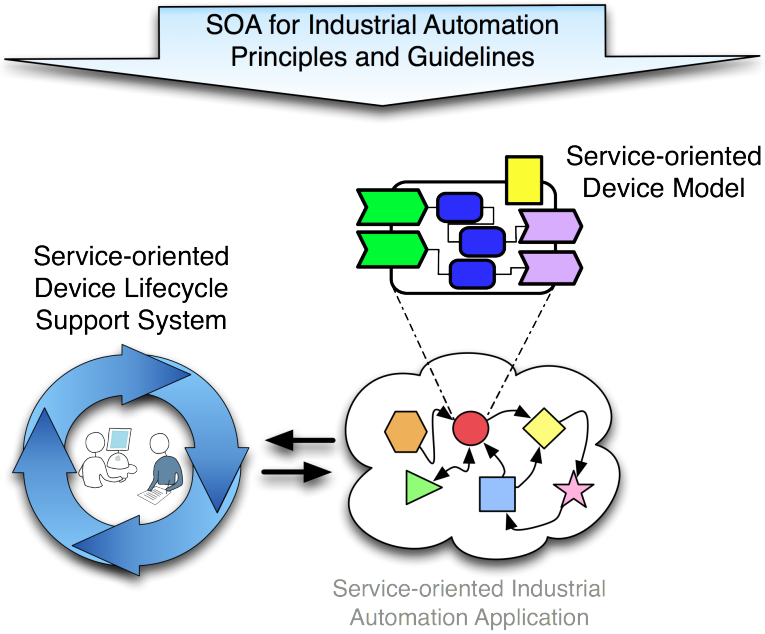


Figure 1.1: Thesis main topics

trigger that can progressively make the domain evolve and feel comfortable with new approaches made to address a still very traditional and closed domain.

As the latest Eurostat reports from [Eurostat, 2012] state, production in EU27 is strongly influenced by social-economical aspects and still suffers with the reallocation of production sites to explore cheap and uneducated labour force along with lower taxes and loose ethical regulation in emergent countries. As seen in Figure 1.2, production indexes in EU27 decreased abruptly with 2008's crisis and had been slowly raising until the current state where the tendency is again inverting to stagnation or even regression. It is visible that the recent production peek is considerably lower than the previous one what supports the idea of slow economic growth or even declining trend. New methods, approaches and business models are then indispensable to invert this trend.

Even with previous fluctuations and uncertainties, the control and monitoring domain worldwide has a market value of 190 billion Euros, as shown in Figure 1.3, where the European market represents around one third of it. Each application segment is split in hardware, software and services. The sum of automotive, manufacturing and process industries represent 60% of the total market, where almost two thirds of this segment relates to services. Integration, installation and training accounts for around 50% of these services following the same study.

As depicted in Figure 1.4, the levels of expected average growth are distinct for each subdomain, but it is still easy to detect a higher trend for software and services alongside some network hardware.

In a more abstract interpretation, it is possible to detect a particular trend related to

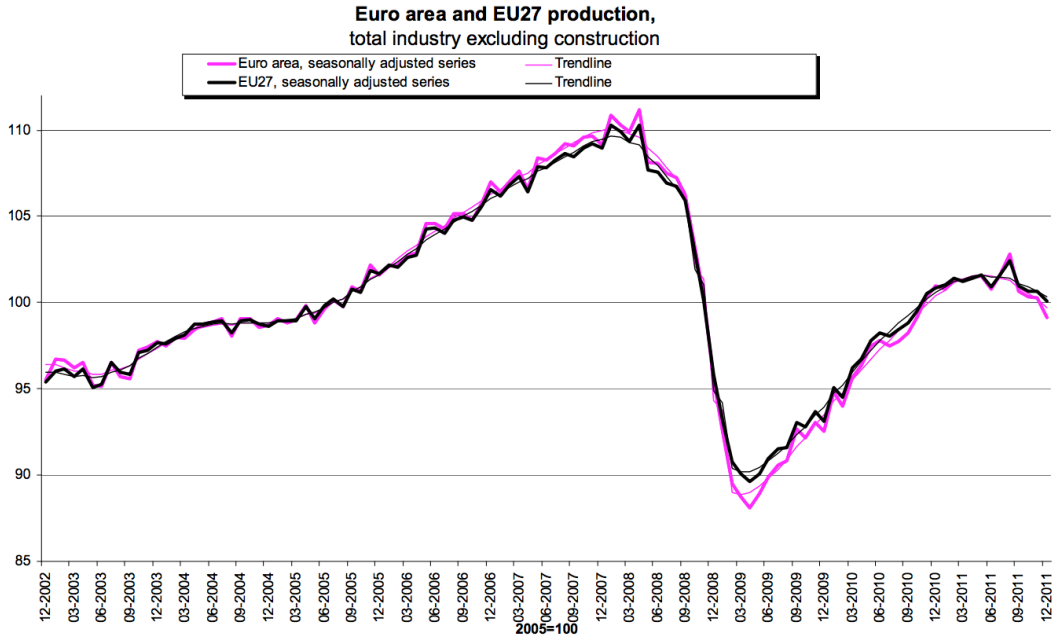


Figure 1.2: Euro Area and EU27 production rates since late 2002 [Eurostat, 2012]

the crescent use of network connected devices that can ease the integration, decision support, management and transparent communication across the different enterprise ICT layers. In fact, the new generation of products is expected to already include some level of service packages and this will strongly influence future devices models and infrastructures. However, the introduction of all this new added value will impose new challenges to technical background and know-how of domain experts. Also, new global factors promise to influence the evolution of the domain solutions such as energy efficiency, cost, security, reliability, openness, environmental regulations, cost of labour, ageing population, etc. With all these challenges and restrictions in mind, production companies are still seeking competitive advantage and reduced costs by attaining increased plant availability and productivity, reduction of down times during product exchange and conflict situations promoting an effortless lifecycle management.

Following actual trends and domain requirements and expectations, the current work as a whole is expected to deliver a significant impact to the service-oriented industrial automation domain. First of all, by presenting a deep analysis about the adoption of SOA to industrial automation, the current work delivers a reference documentation on how to better exploit service-oriented principles, methodology and technology in the current domain. Specific requirements, constraints and expectations of both are compared, merged and updated to better assist the design, development, deployment and management of future service-oriented automation systems.

A service-oriented device model is proposed envisioning an integrated approach to deploy SOA at device level, while at the same time it opens the door for a more transparent interaction across the different ICT enterprise layers. The device model is open

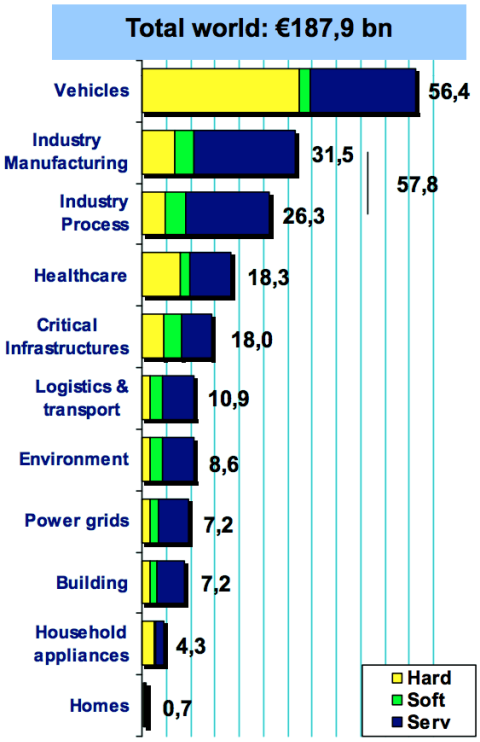


Figure 1.3: Worldwide monitoring & control markets by application [Decision - Etudes and Conseil RPA, 2008]

and adaptive enough to handle the diverse aspects of system lifecycle evolution, even for other domains of application. It plays a major role on the support of the overall system evolution by embedding out-of-the-box services that enable transparent discovery, identification, setup, and management and, at the same time, allow effortless device and services customization. Also, the methodology to deploy and manage resources within a physical device, as well as allowing manufacturers to embed their own built-in services in their devices, is supported by open standard possible to be employed at different levels of the system infrastructure. Although exploiting a paradigm still considered new for this domain of application, this approach is still backward compliant with current field knowledge due to services interface abstraction and clear hardware versus software separation.

Each element of the proposed architecture exposes its skills as services in the network, which will enable a customized composition of modules and a mutual transparent interoperability. This same architecture supports and promotes the exploitation of Internet-based business models in a domain that is traditionally closed to these. As an example, services developers can make their services available online in a web portal so that users can search, find and retrieve the set of services that can fit current scenario. These may include resources that implement common or enhanced functions for control, management, monitoring, security, etc. for generic or particular equipment. The systems

Products, solutions	2007-2020 Average growth rate
Control	2,2%
Interfaces	5,6%
Network	8,0%
Computing systems	4,3%
OS, drivers	4,3%
Total hardware	3,8%
Comm	8,8%
Appli & visualisation	6,9%
Dev., simul. & modelling	11,9%
Decision, support syst, ERP	12,4%
Total software	9,2%
Application design	8,9%
Integrat.,instal. & training	7,8%
Comm & network	10,5%
Others: maintenance	10,9%
Total services	9,4%
TOTAL	**7,8%**

Figure 1.4: World monitoring & control growth by product and application [Decision - Etudes and Conseil RPA, 2008]

integrator simply needs to search the catalogue for the ones that best fit the current application, download them to the local repository and deploy them into the appropriated devices. As example, when built-in services are not enough to handle particular system requirements, there is a need to deploy extra services that might allow a more complex control or management of the device.

This approach can open new business models since any developer can now create its own resources, publish them online and make them available to the community. By promoting competition among service developers, systems integrators and end users will benefit from the added value, robustness, performance and availability of these. Furthermore, the clear separation between hardware and software layers will allow the same service to be deployed into distinct physical devices. Even if implementation changes due to variations on device firmware capabilities, the service interface remains unaffected for the same essential function – a service can be substituted by an equivalent one without changing anything only by keeping the service interface.

Nevertheless, the inverse path, i.e. access to devices and services installed in the shop floor, must be seriously taken into account to avoid security issues and the leak of critical or proprietary data. To face these requirements, several web services standards and initiatives are already encompassing the different aspects of the problem.

By laying over open web standards and focusing on interoperability, modularity, uncomplicated management and reconfigurability, particularly enhanced by the use of the dynamic deployment feature, it is possible to deploy a set of components that together form a customized infrastructure to cope with devices and services lifecycle evolution for industrial automation.

1.5 Input and Contribution to other Research Initiatives

During the execution of the present work, the author was a member of the Centre of Technology and Systems (CTS) at UNINOVA, which develops its activities in a wide range of R&D domains, such as microelectronics, telecommunications, energy efficiency, industrial control and decision support systems and computer engineering. Although the present work can be considered multi-disciplinary, it mainly fits the research vector that covers the subdomains of intelligent control and decision support system applied to industrial automation. This work is also the result of tight collaboration between UNINOVA and Schneider Electric France. During this period the author integrated *DInnov* (Innovation Department) team at Schneider Electric – 38TEC, Grenoble and upon which most of the current work was developed and validated.

The proposed work was developed in close interaction with several European collaborative R&D projects. The early participation in EU's Sixth Framework Program (FP6) EUPASS project [Onori et al., 2008a] provided the author some insight over the shop floor Evolvable Assembly Systems (EAS) concepts. This initial knowledge and experience supported a posterior strong involvement in ITEA SODA [SODA, 2009] and Seventh Framework Program (FP7) SOCRADES [SOCRADES, 2009] projects. SODA mostly focused on the tools and methodologies necessary to deploy a complete SOA ecosystem at industrial automation device level. Exploiting SODA results, SOCRADES covered a broader domain of application comprising the design, execution and management of complex SOA automation systems. In parallel to these, InLife project [InLife, 2008] researched the application of SOA principles and methodologies to device level diagnostics and maintenance employing a Ambient Intelligence (AmI) approach. The FP7 Self-Learning project [Self-Learning, 2012] addresses the industrial production domain and it is supported by a SOA infrastructure. The key assumption here is that a context awareness approach will allow improved adaptation and enhancement of control and other manufacturing activities of production systems, so-called secondary processes such as energy use optimisation, monitoring or maintenance. Moreover, FP7 COSMOS project [COSMOS, 2012] addresses wind turbines sector and researches cost-driven adaptive control exploring the SOA side of the OPC-UA standard. Previous project results also enabled its adaptation to device level of the domain of energy production, transport and consumption in the scope of the ITEA NEMO & CODED project [NEMOCODED, 2012]. Its major objective was to promote and support energy efficiency through the deployment of a SOA based infrastructure compliant with domain standards.

As a short summary, all these projects allowed important knowledge gathering and sharing, while at the same time they provided a fundamental test-bench to work hypothesis and approach. The framework of the current work intersectes the domains of the projects in which the author was involved as depicted in Figure 1.5.

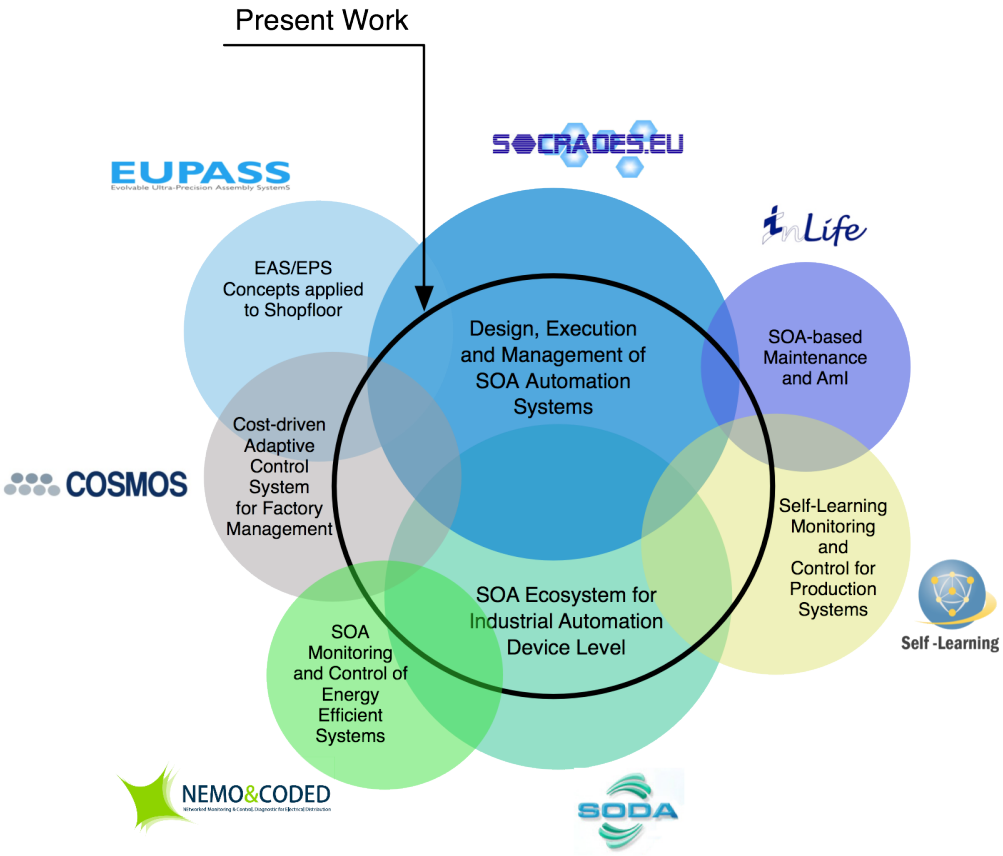


Figure 1.5: Thesis framework within the scope of the participation in European research projects

1.6 Research Methodology

The present research work follows the classical scientific methodology, which includes a set of steps that may lead to the characterization and explanation of a specific phenomenon, as presented in Figure 1.6.

The first stage of this method relates to the formulation of the problem or research question; i.e. what needs to be questioned or explained about the phenomenon. This step emerges from the need to address new emerging information about a phenomenon that either refutes previously established hypothesis or the phenomenon itself is new and there is no scientific knowledge detailing it.

For this particular work the research questions are prepared to focus on the study and evaluation of the application of SOA to modern industrial automation device level, along with the specification of new models, infrastructure and entities to support its life-cycle agile management and evolution. The smooth combination of the previous aspects has not been completely addressed as an integrated holistic approach by the research community and the current work is expected to deliver an important input in this area.

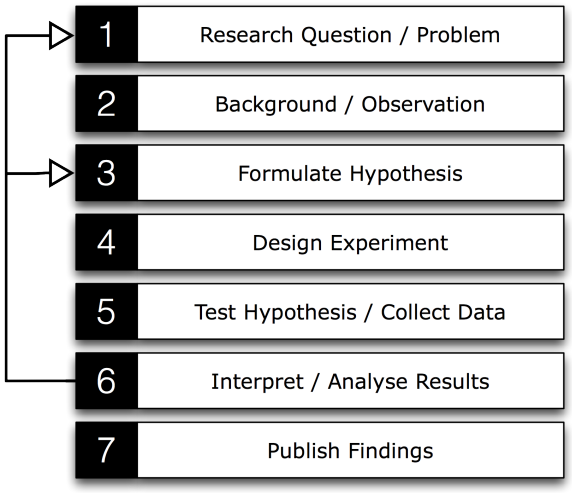


Figure 1.6: Classic research method overview

Taking into account the current status of scientific research process, the domain characterization done by [Egger and Carpi, 2008] reflects the actual process endured along this thesis work (see Fig. 1.7). This last model can be interpreted as the adaptation of the classical research method to contemporary Research and Development (R&D) environment.

Throughout the participation in several collaborative R&D projects and tight interaction with several domain experts, the current state-of-the-art review denotes that there are aspects that still need further work or refinement.

As initially discussed in [Zelkowitz and Wallace, 1998], the process of validating concepts supported by software-based aspects is normally a complex and essentially a qualitative task due to the fact that test-case scenarios can be influenced by implementation options which need to be profoundly identified and assessed. In a more recent survey from the same author [Zelkowitz, 2009], it indicates that the research community is paying more attention to these aspects, particularly by using the proposed taxonomy for validation methods. Following this same taxonomy, section 6 employs field study, synthetic and lessons learned validation models to present work context.

The last step of the trailed scientific method corroborates the contribution of the present work to science and it is also an important step in validation through peer acceptance. In this context, the following international journals (Figure 1.8) and conferences (Figure 1.9) have been identified as preferred dissemination and peer validation mechanisms as they are technically sponsored and supported by established and recognized organizations and domain experts.

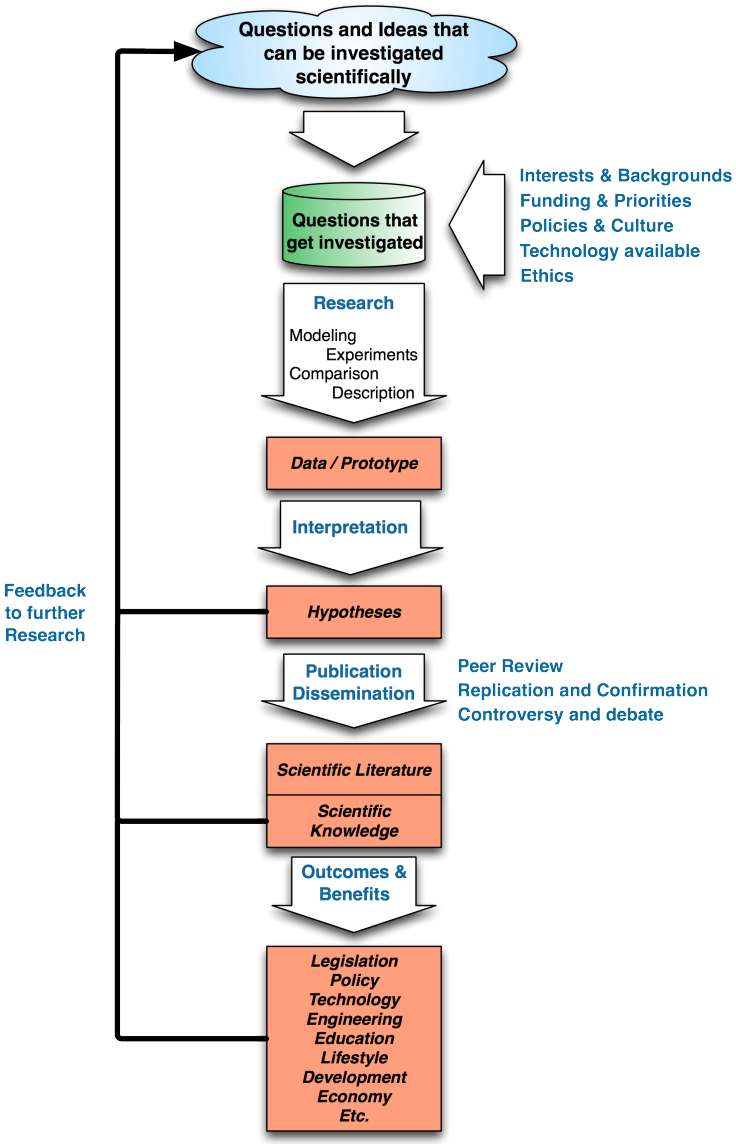


Figure 1.7: Diagram for the Process of Science (simplification of the original from [Egger and Carpi, 2008]

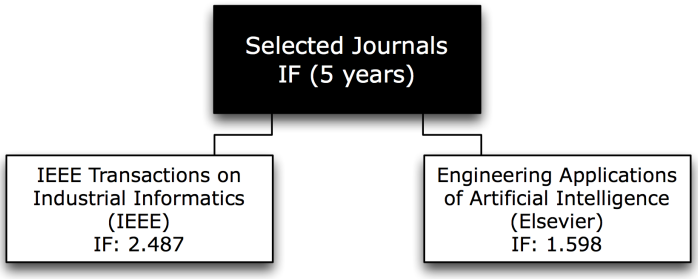


Figure 1.8: Selected international journals

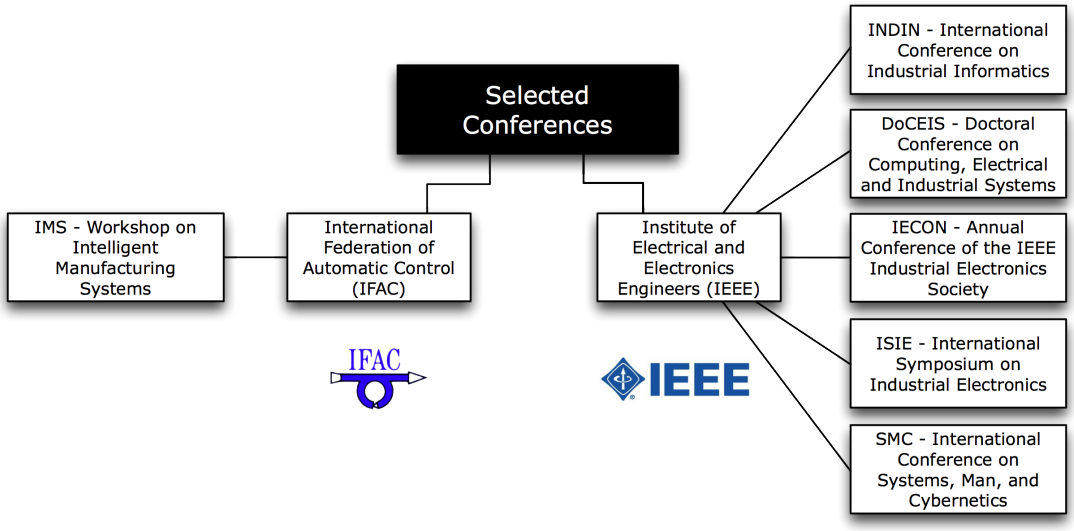


Figure 1.9: Selected international conferences

1.7 Thesis Outline

This thesis is organised in seven chapters:

- The current chapter, i.e. **Introduction**, introduced the background and research problem addressed by the current work, along with the research questions, hypothesis and approach followed to achieve the aimed contributions. Furthermore, it is also discussed the envisioned domain impact associated to current work results, the input and scope of the present work within several research initiatives, as well as the followed research methodology.

- Chapter 2 (**Supporting Concepts**) encloses a summary of the concepts and contextual framework that will be referred throughout the document such as contemporary industrial automation requirements and expectations, paradigms and trends, SOA concepts and methodology, semantic web amongst other related subjects.

- Chapter 3 (**A Survey on SOA for Industrial Automation**) reviews the current state-of-the-art on SOA applications in industrial automation with a focus at device level. This chapter also enunciates the areas where SOA can deliver some vital contributions as well as a set of open challenges and guidelines for an increasing adoption of SOA on the assistance of systems integrators along a service-oriented application lifecycle.

- Chapter 4 (**Device Lifecycle Support Architecture**) presents the proposed reference Device Lifecycle Support Architecture specified to address some of the open research challenges detected on previous chapter. Architecture elements are presented along with the according scope and roles.

- Chapter 5 (**Service-oriented Device Model**) specifies the proposed Service-oriented device model addressing industrial automation device level. Besides the description of model resources and particular categories of services possible of being hosted by devices, a special attention is paid to the deployment service and resource state-model. Taking into account current ascendency of DPWS and OPC UA specifications at industrial automation device level, an assessment of the two is presented alongside WS-Management to deliver a convergence proposal.

- Chapter 6 (**Implementation & Validation**) discusses some aspects related with validation options concerning architectural proposals in this domain. Also, the various implementations done in the scope of this thesis regarding the proposed reference architecture and device model are presented and results discussed along with the accomplished scientific contributions related to the current work.

- Chapter 7 (**Conclusions**) extracts the overall conclusions and matches them with initial research questions and hypotheses counterparts. A set of challenges and constraints associated to the current work results and domain is also presented. To finalize, future work are presented.

2

Supporting Concepts

This section presents some background concepts and contextual framework that will support the main subjects addressed in this thesis. This outline also includes the literature review on these cross-domain aspects and it is divided into three main subsections: Industrial Automation trends, SOA concepts and Semantic Web topics.

2.1 Trends in Industrial Automation

2.1.1 Overview

With the dawn of mass production in the context of the industrial revolution in the later part of the eighteenth century the world has experienced a continuous growth in almost every aspect of the daily life of the average citizen. This growth led to the increasing demand for high quality and highly customized products at low cost as evident in society nowadays, as depicted in [Pine and Davis, 1999]. In the last years almost all industrial automation companies have deeply felt the effects of globalization. These demands oblige companies to cope with a minimum possible time-to-market in order ensure a clear advantage over major competitors preferably without escalating expenses. As exposed by [Feenstra and Hanson, 1996], all this is forcing major companies to push their production sites to underdeveloped zones where cheap and untrained manpower is especially available, and where it is possible to escape from strict labor conditions policies. However, this solution only focuses on retrieving profit by cutting personnel wages and labor conditions – traditional shop floor reality remains barely untouched and do not provide really new solutions to avoid or reduce the production offshoring trend. Modern companies are becoming more aware of their role and impact on future generations and subsequent future market evolution. These are now starting to adopt more sustainable approaches also to improve their image close to their clients. As stated by [Brundtland,

1987], sustainability promotes a "*development that meets the needs of the present without compromising the ability of future generations to meet their own needs*". The combination of the customized demands by the customer with the sustainable development desires by the company will raise some serious challenges on how handle this reality within every enterprise layer.

Agility is a fundamental requirement for modern industrial automation companies to improve sustainability and face challenges triggered by globalization, environmental and working conditions regulations, improved standards for quality and fast technological mutation [Gould, 1997, Lin et al., 2006, Vernadat, 2007]. As defined by [Goldman et al., 1995], agility corresponds to operating efficiently in a competitive environment dominated by change and uncertainty. The notion of agility in this context involves being more than flexible, lean or even reconfigurable. In summary, while flexibility refers to an enterprise that can easily adapt itself to produce a range of mostly predetermined products and lean essentially means producing without unnecessary waste, reconfigurability denotes an ability to tackle uncertainty by relying on reconfigurable components that can be composed to adapt to a new unpredictable situation. However, this reconfiguration ability by default does not take into account cost or delay involved in this reconfiguration process – agility implies being reconfigurable and at the same time efficient, reliable, adaptive and competitive by providing a solution in a worthy time frame and cost.

The overall company agility is always limited by its least agile building block – all enterprise ICT levels, from ERP to field device level, need to be agile and interact in a seamless and synchronized manner. The automation device level plays a fundamental role, since a device is the last frontier where high level process workflows are transformed into a structured collection of physical actions executed by physical equipment. Shop floor device level is characterized by a high degree of diversity in device functionality, form factor, network protocols, input/output features, as well as the presence of many heterogeneous hardware and software components, as depicted in [Colombo et al., 2004, Cannata et al., 2008]. It is then fundamental to specify a unified approach to cope with the complete system lifecycle covering all its different phases of operation.

2.1.2 Industrial automation requirements

2.1.2.1 Introduction

As discussed in [Schoop et al., 2002, Jammes and Smit, 2005b], it is evident that industrial automation is still nowadays a domain with numerous challenges to be addressed both in terms of methodologies and technology solutions. The overall main concerns are:

- Long time between system design and final installation;
- Complex and time-consuming reconfiguration to face product variations or requirements update;

- Extremely centralized/hierarchical approaches lead to common monolithic implementations;
- Scalability involves exponential complexity;
- No default fault-tolerance/redundancy;
- Equipment incompatibility between different vendors and legacy systems;
- Lack of widely accepted standards;
- Shop floor systems are still commonly isolated from higher level business environments.

In reality, the world of industrial automation device level is characterized by a high degree of diversity in device functionality, form factor, network protocols, input/output features, etc, as well as the presence of several hardware and software platforms. In areas with a large base of installed devices like industrial automation, this has often resulted in a patchwork of technology confinements with poor scalability [Colombo and Karnouskos, 2009].

At the industrial automation shop floor, the fieldbus technology covers a very wide spectrum of techniques and solutions. This phenomenon may be the reason why there is so much disparate offer and lack of a real standard. Although it can be considered nowadays an essential component of a big percentage of industrial automation installations, Internet-based technologies are expected to overcome initial suspicion and shortcomings to assume its position against current fieldbus solutions as stated by [Thomesse, 2005]. A vision also shared by [Weaver, 2001], which support the Internet architecture as the preferred solution for remote monitoring and control. As also remarked by [Alves et al., 2000], recent advances in Ethernet such as the Fast/Gigabit Ethernet, micro-segmentation and full-duplex operation using switches will enable a better support to time- and resource-critical applications.

Current industrial automation environments are composed by several heterogeneous network environments that need to be integrated as effortlessly as possible. To avoid these issues, end-users frequently tend to pick a complete line of products from a unique Original Equipment Manufacturer (OEM). This situation severely restrains system openness to integrate new devices, especially from different OEMs, since each company is still much based on their proprietary standards and tools. The integration effort in these cases is costly and the reengineering phase tends to be much longer than initially expected. As stated by [Sauter, 2005], *integration has always been the ultimate goal of automation*.

The next section depicts the contemporary industrial automation domain requirements by subdomains of application.

2.1.2.2 Industrial automation areas

As discussed in [SOCRADES, 2007a, SOCRADES, 2007b], the following subsections introduce the latest trend topics in industrial automation domain.

2.1.2.2.1 Application (Re)engineering

ultimate desire of a production company is to provide the client with the product he needs in the time-frame he wants at competitive price. The reengineering phase is then crucial to attain this goal – the industrial automation systems integrator must be capable of reconfiguring the system easily and fast while guaranteeing a suitable performance. These tasks include the update of each device program, configuration, physical and network topology, range of interaction, among others. At any time, a new device can be added, removed, reconfigured and the existing infrastructure must support it, preferably done at a fast rate. The device hardware plays also an important role on system flexibility, e.g. a device with a hardware built-in application impossible of being reconfigured cannot adapt to a new context not predicted during design phase. The automation application needs to be easily deployed into the devices, either for a centralized or distributed approach. This is a particular difficult task due to the wide range of incompatible proprietary tools and deployment methodologies. Note that before restarting the complete production system or part of it there is a need to debug and validate the changes made, in addition to a possible troubleshooting to restore a faulty process.

Other hot topic is the independence of the software application from the target device, i.e. hardware platform. The application code should be portable so that it can be deployed into several heterogeneous devices, with none or just a few variations on the configuration. This implies a flexible code deployment tool capable of discovering and adapting to the target device.

2.1.2.2.2 Services Composition

In order to provide added value over the different elements that constitute the environment, different devices capabilities (in this context, most times referred as services) need to be aggregated, at different levels of abstraction, from low level control devices (sensors and actuators) to higher level Supervisory Control And Data Acquisition (SCADA), Manufacturing Execution System (MES) and ERP applications. The so called atomic services, hiding implementation details, need to be straightforwardly assembled into more complex ones and so on, as in the "Russian doll" concept, rising the service abstraction as referred by [Jammes and Smit, 2005b]. By rising service abstraction, high level process descriptions become available to be used by the application without the need to regard low level intricacies. Instead of having implicit addressing of a function, e.g. the change of a vague output variable value, a service should be discovered based on the functionality it provides and exposes; i.e. services should be self-describing and self-contained.

The ultimate objective is to turn assembly of services automatic or at least more independent from human input. Although this vision is mostly directed to application design

phase, this dynamic behavior can be even step-wisely applied to run-time phase making a device capable of discovering, choosing and interacting with the service that best fits its current needs; e.g. during an interconnected device down time due to a maintenance intervention or failure.

2.1.2.2.3 Self-* ability

The so called devices self-* ability is increasingly a hot topic allied to the promises of autonomous and still interoperable intelligent devices running on future industrial automation installations and on many other domains as discussed in [Di Nitto et al., 2008]. Self-* ability refers to the set of skills that a device embeds to make it autonomously adaptive to the current context

The tendency for processors to become increasingly powerful, smaller, and cheaper leads to the ubiquity of tiny intelligent devices. This tendency is already a reality in many domains of application, such as home and building automation, power distribution, multimedia, etc. Since extensively manage and control an environment full of devices will become an oversized task, these devices will need to be more autonomous by having local decision control, preventive maintenance, diagnosis and even some level of "self-healing" capabilities. The service is considered autonomous since it is created and operated independently of its environment providing self-contained functionality, i.e., this functionality would be useful even if it is not associated with any higher-level system. Even embodying some level of autonomy, a device must be interoperable so that others can exploit its skills.

2.1.2.2.4 Lifecycle Support

Along the self-* ability, lifecycle support features are also long-awaited to be applied from field to higher level scale, regarding both application and device scopes. While at application level the parameters to manage are normally too specific to that particular case, the device itself can already embed some generic services that can allow systems integrators to manage and control it in a standard manner from the first instant it is taken from the box. These intelligent devices should provide built-in services for setup, control, management, monitoring and diagnosis. The way to use these same services should follow SOA-based standards widely accepted by the community. One of the major clients complains when dealing with a new device is exactly how to interact with it at the start just to define simple parameters (e.g. Internet Protocol (IP) address) or check the current device status and configuration parameters.

2.1.2.2.5 Device Capabilities

The wide collection of available devices with diverse capabilities and different domains of application makes the choice of the most suitable device sometimes tricky. Consequently, the platform should be also compatible with devices with low resources. These

low price devices are well suitable to perform as autonomous entities in a Sensor/Actuator network. New automation approaches must provide, at its smallest set of features, capabilities already available in commonly deployed technologies. Reliable and real-time messaging, in addition to "fieldbus-like" features can be mandatory requirements in automation environments. A bridge between wireless and wired networks is also an added-value feature.

2.1.2.2.6 Enteprise-wide integration

The integration of industrial automation devices within the above enterprise ICT platform is becoming a key factor to endorse the overall management and control over a complete business environment most of the times distributed over different levels of organization and even geographical areas. Seamless and effective ICT integration is then a major goal. The effortless access and management of distributed data is still a more complex subject. The ICT platform should automatically detect the introduction or the removal of a device in the network. This device should also provide a location awareness feature, so that it would possible to find it on the environment and, most important, be sure that it is the one to be addressed.

The summary of requirements of the industrial automation domain from the end users point of view and system integrators is depicted in Table 2.1, while Table 2.2 summarizes user requirements for application systems engineering and lifecycle support.

2.1.2.3 Main challenges

Taking all these factors into account, the current main challenges are strongly related with fact that **an industrial automation environment can be considered a distributed network of heterogeneous devices**. First of all, every device must be easily reachable by allowing a straightforward discovery and identification, as well as configuration and management of the existing resources. This subject is attached to the necessity to integrate shop floor level into the remaining layers of the enterprise ICT infrastructure. Nowadays these domains are mostly disconnected – different paradigms, processes and contrasting technology solutions; i.e. shop floor devices are not easily connected to high-level enterprise applications

To conclude, it is clear that **each industrial automation application requires a customizable solution to cope with device heterogeneity in terms of scope, features and processing power as well as with scenario specificity and requirements**. For each scenario, the **system integrator should compose the application using the set of tools and services that best fit current requirements and that can ease and assist his work along system lifecycle**. Due to the domain traditional methods and technology background, the lifecycle support environment should take into account these disparate domains of expertise and combine them smoothly to enable a unified service-oriented deployment.

Subject	Details
General	Support for heterogeneous network environments Device services aggregation Hardware- and platform-independent design phase Easy access and management of distributed data
Distributed Control	Dynamic assembly of services High level services through aggregation of existing ones
Services Deployment	Easy binding reconfiguration
Management	Notification of new or removed services/devices Get Status/Start/Stop/ Get Properties functionalities Add/remove/stop/reconfigure devices smoothly and fast at any time Automatic discovery/interaction between devices
Enterprise Integration	Devices and IT applications should be able to interact without intermediaries
Wireless Sensor/ Actuator Networks	Autonomous devices at low price Attribute based addressing Location awareness Self-* ability Bridge between wireless and wired networks
Generic Services	Diagnostic web services Integrated monitoring Transparent setup Services deployment Debugging & Troubleshooting of a distributed application
Non-functional	High level process descriptions
Platform requirements	Low resources devices compatibility Reliable and improved messaging, in addition to fieldbus-like capabilities Quality control guaranteed on wireless networks

Table 2.1: End User and Systems Integration Requirements

Subject	Details
Design and Configuration	Ability to configure machines built from smart modules Reuse of machine components High level machine configuration High level process description Plant layout support Distributed system configuration and management
Commissioning and Operation	Ability to monitor products and processes Error management High flexibility for manual interaction Intuitive user interface to control/monitor systems Access to user interface for decision support in products planning, implementation and operation phases
Virtual Engineering	Support for remote assistance Lifecycle support from engineering tools Support for globally distributed engineering teams Access to real-time simulation of systems components
Standardisation and Openness	Open system architecture and vendor neutral systems Effective and seamless IT systems integration Inherent support for compliance with standards Access rights must be clearly specified and controlled

Table 2.2: User Requirements for Application Systems Engineering and Lifecycle Support

2.1.3 Trends

In the context of industrial automation several initiatives have helped steering the progress and provide guidelines and solutions for the upcoming deployments supported by the lessons learned from earlier advances. The next subsection introduce some of the latest industrial automation paradigms that frame the work done in the scope of this thesis as well as an overview of the latest trends in the domain concerning device level communication and control.

2.1.3.1 Industrial Paradigms

As defined by [Bogdan and Biklen, 1982], the term paradigm can be defined as "*a loose collection of logically related assumptions, concepts, or propositions that orient thinking and research*". Especially in the production and manufacturing domains, different paradigms have emerged to tackle the several known issues associated to existing industrial automation installations.

As referred before, despite being programmed to be robust and predictable, equipment and applications fail while changes on requirements imply a continuous need for reengineering in existing production systems deployments. For this reason, uncertainty must be taken into account and reengineering processes need to be sufficiently agile to cope with the evolution of production requirements. Industrial automation applications are today still mostly structured in a rigid pyramidal hierarchical approach, which also compromise the ability to perform relevant lifecycle revisions from top business layers to shop floor device level.

The Flexible Manufacturing Systems (FMS) [Stecke, 1985, Sethi and Sethi, 1990] paradigm tried to anticipate partial system adjustments by providing a priori several built-in features controlled by reprogrammable equipment. Nevertheless, due to its limited flexibility even when dealing with known requirements and inability to efficiently handle new and unexpected requirements led to gradually being surpassed by other paradigms: Reconfigurable Manufacturing Systems (RMS) [Koren et al., 1999, Mehrabi et al., 2000, Bi et al., 2007], Bionic Manufacturing Systems (BMS) [Ueda, 1992, Okino, 1993], Holonic Manufacturing Systems (HMS) [Christensen, 1994, Valckenaers et al., 1997, Van Brussel et al., 1998, Leitão and Restivo, 2006], Collaborative Automation [Gorbach and Nick, 2002, Colombo et al., 2005, Harrison and Colombo, 2005]. and more recently Evolvable Assembly Systems (EAS) [Onori, 2002, Frei et al., 2007, Onori et al., 2008b] and the subsequent Evolvable Production Systems (EPS) [Barata et al., 2007a,Lindberg et al., 2007, Ribeiro et al., 2010].

The RMS paradigm, by including principles of modularity, integration, flexibility, scalability and adaptability emerged as a reaction to FMS shortages As described in [Brehmer and Wang, 1999, ElMaraghy, 2005], RMS endorsed generalized flexibility on demand in a more agile manner. RMS promised a rapid change in structure, but also in

hardware and software components, while a FMS machine is pre-built to perform different type of actions, including some that are not necessary to current scenario.

The BMS paradigm retrieved its inspiration from biologic bodies that are composed by *organs* that interact seamlessly in a hierarchical flow of DNA-like information. When applied to the manufacturing realm, each entity (*modelon*) will evaluate current process and select what other entities are required to accomplish the present task. BMS also employs learning techniques, such as genetic algorithms, to optimize this set of chosen *modelons* in the scope of system design and execution. In the HMS paradigm, instead of *organs* the authors inspired from [Koestler, 1976] endorse the concept of *holon*. Each *holon* can be a part and a whole at the same time while being part of group referred as *holarchy*. Each *holon* will pursue a particular task or goal through coordination, cooperation and negotiation within a *holarchy*. Some holonic manufacturing implementations are already available: PROSA [Van Brussel et al., 1998], PABADiS [Luder et al., 2004] and ADACOR [Leitão and Restivo, 2006]. In some works, the HMS paradigm is associated to Collaborative Automation principles as in [Colombo et al., 2004]. Collaborative Automation results on the blend of three main research axis: Intelligence, Control and Mechatronics applied to the industrial automation domain. Collaborative Automation mostly relies on technical aspects to improve the deployment of more intelligent automation systems based on Multi-Agent System (MAS) or SOA technology as ICT solutions allied to the mechatronics embedded value.

Evolvable Production Systems (EPS) was born out of the foundations of EAS paradigm and was also inspired by previous paradigms from which it absorbed certain strengths and tried to solve some drawbacks while emerging as a broader approach for a wide range of specifications, design, and technical considerations. An EPS is an open system composed of low-granular intelligent devices, in which they can be combined into several forms and collaborate between them to accomplish production goals. Adopting also a decentralized approach control, this paradigm promotes a fast reconfiguration of production aspects following current business opportunities. The system ability to quickly adapt to overcome an existing disturbance by proposing an alternative shop floor setup represents one of the key aspects addressed by the paradigm. Subsequently, the evolution derives from the emergence of the result of these adaptations as a whole. Evolution is expected to be mediated by a top level strategic decision that will result in the development of one or more suitable modules or product reengineering to, iteratively, further explore alternatives in system reconfiguration. EPS promotes reconfiguration instead of system reprogramming. One of the first control architectures to support these features was CoBASA [Barata, 2004], focusing on rapid shop floor reconfiguration by using a contract system to form coalitions of equipment instances to provide higher-order skills.

Although these paradigms tried to tackle critical issues in the context of contemporary industrial automation systems following several lines of investigation and background, there is crescent tendency to push some high-level ICT approaches and technology into automation devices as depicted in the work from several authors [Jammes and Smit,

2005b, Karnouskos et al., 2007, Colombo and Karnouskos, 2009]. In fact, it is a natural path from traditional logic based control into state-of-the-art ICT technology and integration platforms. With the progressive evolution of the technology and availability of development platforms, new options become available to deploy improved solutions that can support the deployment of features and functionalities previously unconceivable for the industrial automation. The systems integration community demands for new tools and services to assist his daily work and reduce the effort, cost and delay during lifecycle interventions. In this context, the industrial automation device level plays an essential role on delivering the estimated levels of agility.

2.1.3.2 Ethernet and IP ubiquity

Firstly published in 1976 [Metcalfe and Boggs, 1976], Ethernet technology rapidly emerged as the unquestionable mean for interconnecting devices in a wired Local Area Network (LAN) as discussed in [Shoch et al., 1982]. This situation was even more evident after it being standardized as IEEE 802.3 [IEEE, 2011]. As described in [Decotignie, 2005, Decotignie, 2009], the use of Ethernet in industrial automation domain, the so-called Industrial Ethernet (IE), is not really new, but it still suffers from the fact that it cannot be considered entirely deterministic. Ethernet was not originally designed with industrial applications in mind and does not entirely fulfil classical industrial automation requirements.

This issue triggered the development of protocols that promised to offer deterministic assurances, such as maximum transfer time or packets lost, jitter not exceeding a predetermined threshold or some guaranteed bandwidth. The so-called fieldbus technology emerged in the form of several solutions, as discussed in [Thomesse, 2005, Neumann, 2007]. These solutions focused on allowing a higher level of Quality of Service (QoS) by enhancing the original IEEE 802.3 with new standards to support network traffic prioritization, switches and clock synchronization. These solutions were developed having the hard requirements of real-time behaviour, functional safety and security. Nevertheless, the large majority of fieldbus proposals presented a poor modularity and encapsulation features, they are not mutually compatible and oblige to software and even hardware modifications. Although some initiatives have pursued a single solution, the practical result is still a patchwork of solutions from different vendors that do not interoperate. These incompatible solutions usually require custom hardware platforms and only support custom device access methods. This situation leads to complex integration implementations.

Moreover and as highlighted by [Neumann, 2007, Beran et al., 2010], the next generation of industrial communication networks must also support the supervision and control of several decentralized small installations; wireless communications; remote access for control, configuration and maintenance; as well as machine-readable knowledge including local awareness. In [Moyne and Tilbury, 2007] the authors foresee a crescent

use of Ethernet-base technology, and also wireless network along the different layers of an industrial automation enterprise driven by its low cost, widespread availability of solutions, tools and Internet compliancy.

On top of Ethernet domination, the IP emerged as the primary communications protocol of the Internet protocol suite. Originally referred in [Cerf and Kahn, 1974], it progressively evolved into IPv4 [Information Sciences Institute, 1981] that supported the world wide spread of Internet. More recently IPv6 [Deering and Hinden, 1998] specification emerged mostly as a solution for a lack of available IPv4 due its larger address space, but also by including a new packet format that simplifies routers processing, improved security and privacy aspects, multicast and extensions support as discussed in [Huitema, 1996, Stallings, 1996]. An broader historic overview can be found in [Leiner et al., 2009].

In the industrial automation domain, IP-based protocols are now extensively employed as described in [Brooks, 2001, Davies, 2007]. Inclusive, some new tendencies are pushing IP applications in industrial automation to the wireless domain. As envisioned by [Gungor and Hancke, 2009], it is fundamental to enable wireless sensor networks to integrate IP architecture so that it can communicate with broader networks such as Internet to query and deliver services. This author as well as others such as [Mulligan, 2007, Hui and Culler, 2008a, Hui and Culler, 2008b, Delsing et al., 2011] recognize IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) [Shelby and Bormann, 2011] as the major initiative to push IP technology to low-power wireless networks using IPv6. Recent developments by [Sleman and Moeller, 2008, Moritz et al., 2012, Samaras et al., 2013] already pushed the 6LoWPAN approach by enabling web services on top of this protocol.

2.2 Service-oriented Architecture

2.2.1 Introduction

Since its original reference in [Schulte and Natis, 1996], Service-Oriented Architecture (SOA) has grown to a broader *de facto* solution for designing, deploying, integrating and managing distributed applications. Originally oriented for business and enterprise level ICT, there is very little doubt that SOA is already a major topic in many society and economy sectors as pointed out in [Krafzig et al., 2005, Rosen et al., 2008, Bell, 2008]. Its influence and adoption are spreading across several disparate domains of applications, such as business platforms, telecommunications, healthcare, transportation, and building, home and, of course, industrial automation. Even for scientific research, SOA has the potential to increase individual and collective scientific productivity by making powerful information tools available to the community as discussed by [Foster, 2005].

Several SOA definitions are available varying in accordance to authors' background and domain of application. As defined by [Erl, 2005] "*SOA establishes an architectural*

model that aims to enhance the efficiency, interoperability, agility, and productivity of an enterprise by positioning services as the primary means through which solution logic is represented in support of the realization of strategic goals associated with service-oriented computing". As remarked by [Josuttis, 2007], "*SOA is not a concrete architecture: it is something that leads to a concrete architecture (...) It is an approach, a way of thinking, a value system that leads to certain concrete decisions when designing a concrete software architecture*". Although being far-reaching definitions they are still much related with enterprise level branch of SOA. Since this work is focused on the industrial automation device level, the author endorses the definition presented in the pioneering work by [Jammes and Smit, 2005b]:

"SOA is a set of tenets for building autonomous yet interoperable systems"

This last definition although simple and concise it tries to show that SOA promotes the smoothly combination of autonomy and interoperability which are, by definition, conflicting concepts. A service is considered autonomous since it is created and operates independently of its environment and it is self-contained. It is also interoperable through its interface that exposes its functionality to the environment abstracting implementation details. In a SOA application, the interface and implementation aspects of a service are uncoupled to enable a holonic-like approach. SOA is, by definition, a platform-, language- and implementation-agnostic approach.

2.2.2 Key principles

In short, services involve two parties: a provider that exposes and supports the service, and an invoker that uses the services while pursuing its own goals. An elementary SOA application consists of services that provide service descriptions and communicate via messages. These components (services, descriptions and messages) form the core of every SOA application. As depicted in the current reference definition for SOA, it is encouraged for services to exist autonomously but yet not isolated from each other. Services are still required to conform to a set of principles that will allow them to evolve independently while ensuring common ground and standardization to achieve interoperability.

A service will encapsulate some logic within a particular context and offer it to the system through its interface. Other services or programs that obey the service contract exposed by the service provider can then employ these services to achieve their own goals. These interactions are commonly pursued through messaging. As enunciated by [Erl, 2005], the key principles of SOA are:

- *Loose Coupling*: relationship that minimizes dependency and only requires that services retain an awareness of each other.
- *Service contract*: communications agreement commonly based on open web standards as described in service description (also known as service interface or contract).

- *Autonomy*: local control over the logic a service encapsulates.
- *Abstraction*: logic and resource minutiae are hidden from the rest of the system.
- *Reusability*: complex logic can be divided into different atomic services that can be later composed and recycled in other applications.
- *Composition*: services can be coordinated and assembled to form composite services.
- *Stateless*: services do not retain information specific to a particular activity.
- *Discoverability*: services should be outwardly descriptive to be found and accessed through discovery mechanisms.

2.2.3 Service-oriented Computing and Service-oriented Architecture

As explained by [Singh and Huhns, 2005, Tsai et al., 2006a, Papazoglou et al., 2007], the Service-Oriented Computing (SOC) paradigm refers to the set of concepts, principles and methods that characterize computing within SOA in which software applications are built based on independent services with standard interfaces. SOC is the computing paradigm that utilizes services as fundamental elements for developing applications. These provide the means to implement a new architecture that reflects the trend towards autonomy and heterogeneity while ensuring resources interoperability. SOC adds the ability to build on conventional information technology in a standardized way, so that tools can facilitate the practical development of large-scale systems. According to [Tsai et al., 2006a], the reason of having both SOC and SOA is "*to explicitly separate software engineering from programming, to emphasize on software engineering, and to deemphasize on programming*". SOC splits software development into three independent groups: Application builders (by software engineers), service providers (by programmers), and service brokers (joint effort from standard organizations, computer industry, and government).

The application of SOC is currently mostly executed using web services platforms. The web services community has created standard interfaces and protocols that allow developers to encapsulate their functions and tools as services so that clients can access them without knowledge or awareness over implementation details. In this context, a service is a software entity identified by a Uniform Resource Identifier (URI) whose service description and transport protocols are preferably based on open web standards. Interactions between web services typically occur as Simple Object Access Protocol (SOAP) messages with the content codified in eXtensible Markup Language (XML) format. Interface descriptions (also known as services contracts) are normally defined in a Web Service Description Language (WSDL) document and clearly define in a XML-based syntax all the required information on how to use that service. The Universal Description Discovery and Integration (UDDI) standard defines a protocol for directory services (also known as service brokers) that contain web service descriptions. UDDI is the traditional form that

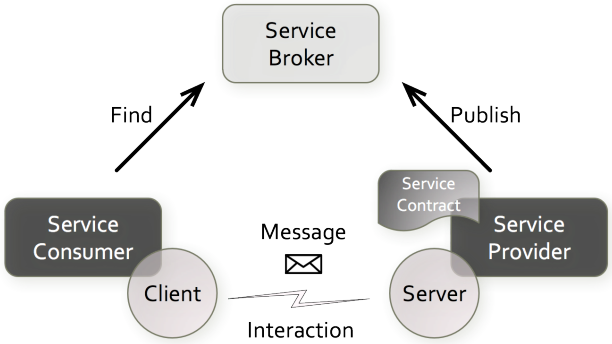


Figure 2.1: SOC classic elements

web service clients have to find services and retrieve their details. Fig. 2.1 presents the relations between these elements.

As backed by [Bichier and Lin, 2006, Papazoglou et al., 2007], SOC endorses the goal of assembling system resources as a network of services that can be loosely coupled to create flexible and more dynamic processes integrating unrelated organizations and computing platforms. SOA is then the path to realize this vision. By providing principles and guidelines to design distributed service-oriented systems, a well constructed SOA installation can enhance the agility level of an organization by delivering self-contained, interoperable and reusable application functions as services and providing a foundation for leveraging these services.

Although SOA approach might not be utterly new, it addresses the fundamental challenges of open and distributed systems in a broader and multi-domain perspectives when comparing with SOC more technology-based domain. This way it envisions the efficient operation and coherence to cope with components autonomy and heterogeneity along their lifecycle as discussed by [Huhns and Singh, 2005]. By requiring that policies are made explicit, a SOA environment can organizationally enforce compliance with these policies and consequently simplify the systems management.

As depicted in [Papazoglou and Georgakopoulos, 2003], SOC domain can be stratified into three layers as in Fig. 2.2: basic service capabilities provided by middleware infrastructure, service composition from system management and business services from system-centered services.

As shown by [Paul, 2005] all major ICT companies, including BEA, IBM, Microsoft, Oracle, HP, SAP, Intel, Cisco, Juniper, SAP, and Sun are now supporting the SOC paradigm. Some of these companies have collaborated to develop languages, protocols and standards to be used in SOC applications. The names of these specifications generally begin with "WS-", so the group of them is commonly referred to as WS-*. Organizations as the Internet Engineering Task Force (IETF) [IETF, 2012], the World Wide Web Consortium (W3C) [W3C, 2012] and Advancing Open Standards for the Information Society

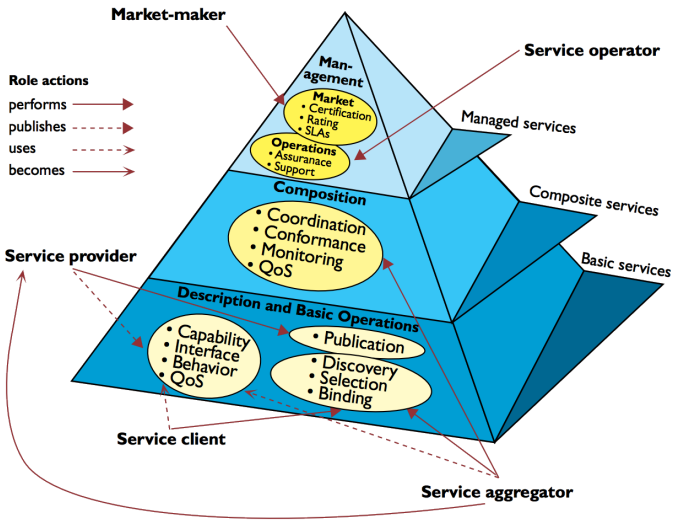


Figure 2.2: SOC research roadmap [Papazoglou and Georgakopoulos, 2003]

(OASIS) [OASIS, 2012a] are the most renowned standardization bodies in this domain. The Web Services Interoperability organization (WS-I) [OASIS, 2012b], now part of OASIS, acts as a regulator. It encompasses a diverse group of experts and organizations to promote best practices on using web services technology by defining profiles and testing tools for the existing WS-* standards. An interesting poster overview on available web services standards is available at [innoQ, 2007].

In the scope of this document, the author will refer to SOA along the text taking into account the fact that SOC is considered to be a part of a broader and more generic concept that is SOA.

2.2.3.1 SOAP or REST?

When analysing the majority of the literature concerning SOA deployments, two implementation specifications arise from the rest: SOAP and REpresentational State Transfer (REST). As described by [Curbera et al., 2002, He, 2003], a Simple Object Access Protocol (SOAP) web service is the most common form of web service in the industry nowadays. SOAP acts like an envelope that carries message contents over diverse transport protocols. It allows rich Message Exchange Pattern (MEP) ranging from traditional request/response to broadcasting and advanced message correlations.

A REST web service (sometimes referred as RESTful) [Fielding and Taylor, 2002, Richardson and Ruby, 2007] is inspired by the concept of "resource" and emerged as a response to the heavier SOAP-based standards. A resource is anything that can be identified by a URI. REST requires less infrastructure support apart from standard Hypertext Transfer Protocol (HTTP) and XML processing technologies. REST web services interfaces are supported by HTTP-based operations (GET, DELETE, POST, PUT), messages are usually in XML based on a XML Schema, and simple messages can be even encoded directly in the Uniform Resource Locator (URL).

In a short comparison, SOAP is a robust and flexible protocol for XML-based distributed computing particularly for critical or legacy systems while REST is more suitable for light and simple web applications. Also, REST can be considered an architecture while SOAP is a protocol – it is possible to send SOAP envelopes in a REST application. As concluded by [Pautasso et al., 2008], RESTful services should be employed for ad hoc web integration while WS-* services suits professional application integration scenarios with longer lifespans and advanced QoS requirements.

Although both approaches are language and platform agnostic, SOAP is also transport agnostic. By being the prevailing standard to implement web services, SOAP has also a wide offer of tools and extensions in the form of WS-* standards, it offers a standard client-consumer contract that improves type handling (WSDL), it includes built-in error handling and it supports message attachments. On the drawbacks side, SOAP is conceptually more complex than REST, it is more verbose what implies an extra cost in time and processor to parse messages and it might be harder to develop if not using the tools already available.

Regarding REST, it is easier to develop, lighter due to the inexistence of an additional messaging layer, it has a small learning curve, it is more independent from tools and it supports data formats other than XML. As investigated by [Mulligan and Gracanin, 2009], REST proved to be more efficient in terms of network bandwidth utilization and round-trip latency in Remote Procedure Call (RPC)-like interactions. REST disadvantages include simple point-to-point communication model (do not support messaging through intermediaries), lack of standards to support security, reliable messaging, policy, etc.; imply harder client-side developments as system requirements become more complex, strongly tied to the HTTP protocol, no agreed way to do versioning and messages are often inconsistent due to the lack of service contract and subsequent verification.

As previously stated, SOA is a conceptual idea while SOC refers to the implementation aspects that can support it. This way, both SOAP and REST solutions can be employed to develop SOA applications.

Due to the critical nature of industrial automation, SOAP-based approaches have revealed to be more suitable as stated in the work by several authors as [Jammes et al., 2005a, Colombo et al., 2005, Decotignie, 2005, Karnouskos et al., 2007, Delamer and Lastra, 2007, Jammes, 2011].

2.2.3.2 Enterprise Service Bus

As originally defined by [Chappell, 2004], "*Enterprise Service Bus (ESB) is a standards-based integration platform that combines messaging, web services, data transformation and intelligent routing to reliably connect and coordinate the interaction of significant numbers of diverse applications across extended enterprises with transactional integrity*".

Other authors as [Luo et al., 2005, Schmidt et al., 2005] also present and endorse ESB as the infrastructure that can leverage a fully integrated and agile end-to-end SOA by

providing connectivity layer between services. ESB does not comprise business logic of services, service clients or containers: ESB basically deals with the meta-data. This meta-data is stored in a ESB broker to assist the process of mediation and matching between service clients and providers. All metadata should be able of being discovered, retrieved, and modified along system lifecycle. A link identifies the "ideal counterpart" between these two realities. They can be configured or dynamically retrieved through a run-time match between service requirements and available capabilities. To achieve this goal the ESB requires a clear definition of interfaces, abilities and requirements.

As presented by [Bichier and Lin, 2006, Menge, 2007] many vendors already adopted this approach to facilitate the integration of enterprise software components in a wide-area enterprise environment. An ESB emulates a flat and transparent network comprising both tools and resources that communicate supported by middleware solutions that intrinsically offer security, distributed transactions and accounting employing message-based open standards. As also remarked by [Papazoglou et al., 2007], the main goal of ESB is to loosely couple system components by breaking integration logic into singular and easily managed resources. The ESB approach is a mean to achieve the vision of Software-as-a-Service (SaaS) depicted by [Gilart-Iglesias et al., 2006a]. SaaS is a business model that promotes software delivery to the end user as a service instead of packaged software without requiring local installation.

In the context of the present work, the ESB is an important cornerstone to provide more complex behaviour and allow end-to-end transparent and fluid interaction across the several enterprise ICT layers. In terms of gains, ESB can increase system agility when addressing requirements modifications, scales to enterprise-wide deployments, promotes configuration over programming and it does not endorse any central regulatory entity. On the counterpart an increased overhead should be expected due to the introduction of one more interoperability layer.

2.2.3.3 Object-oriented dissimilarity

When discussing SOA aspects there is sometimes a misguided comparison with Object-Oriented Programming (OOP) approaches. Many authors such as [Woods and Mattern, 2006, Tsai et al., 2006a, Erl, 2007] already discussed this subject. It is true that SOA, or more particularly SOC, evolved from the object-oriented computing paradigm, but there are substantial dissimilarities between these two paradigms that should be clarified:

- *Methodology*: In OOP approach, the application is based on tightly coupled classes usually organized in a hierarchical structure embracing inheritance relationships. In SOA the developer identifies loosely coupled services and composes them to build applications.

- *Reuse*: In OOP, code can be reused through inheritance and library functions, however they need to be imported for compilation and are platform dependent – to

communicate two applications must be written in the same programming language. In SOA the reuse of code is at service-level – users only need to know their interface to integrate them into the current application. Services by definition are platform independent, can be discovered and remotely accessed.

- *Binding and Composition*: In a OOP, each method has to be linked to executable code before the deployment of the application. A service can be discovered and used in the currently running application.

- *State management*: OOP is more suitable to implement a stateful application while SOA is more indicated to support a stateless environment.

- *Maintenance*: Whenever there is a need to perform updates to an OOP solution the whole application needs to be stopped to execute the modifications. Due to the distributed nature of SOA it is possible to update some systems parts leaving the rest of system running as normal. Besides, even if a service provider updates a service implementation without changing its interface the whole application will remain untouched.

- *Abstraction and Cooperation*: The development of OOP is often assigned to a single team which is responsible for the application lifecycle support. Developers must have the knowledge of the domain and programming. In a SOA the development is usually delegated to service providers (who code the service implementation underneath a service interface), application builders (who compose the application logic based on the available services) and service brokers (who help application builders finding the required services).

As cleverly remarked by [Josuttis, 2007], "*OOP paradigm is a programming paradigm for applications, while SOA is an architectural paradigm for system landscapes*". OOP does not scale and SOA is the approach to integrate systems written in a OOP or other programming technique. These two approaches are not concurrent but complimentary – both OOP and SOA design and development approaches have their place in contemporary integration procedures in different scopes of application.

2.2.4 Governance & Versioning

Even in a well designed and prepared solution infrastructure it is sometimes impossible to entirely anticipate everything from the start. As bigger the system becomes, less static it turns out to be. Either due to new requirements, new implementation methodologies, or simply periodic updates, a SOA system as well as the majority of distributed systems needs to be able to easily and effortlessly evolve. The major issue here is how to ensure the correct balance between maintenance and innovation as discussed by [Erl, 2007].

An important topic in SOA domain concerns governance [Schepers et al., 2008]. Governance addresses the subjects encompassed by the need to ensure that what infrastructure developers and contributors do what it is right when it is needed. It is proportionally imperative to deploy governance roles and mechanisms as bigger as the SOA environment is in order to avoid system chaos and waste of time and resources.

As presented by [Josuttis, 2007], SOA governance can be decomposed into four topics:

- Policies that define what is considered adequate to do.
- Processes that enforce policies.
- Metrics that expose and verify policy enforcements.
- Organization that establishes an environment that enhances the governance process.

Besides all the efforts made to employ a good governance strategy, the reality is that most of the times the idealised project is still far away from the implementation experience. The official enterprise-wide process is all too often never entirely implemented, the majority of the developers do not really follow it in detail, and sometimes they do not even know about its existence. Afterwards, there is the perceived process that consists of what company common sense observes about the current application behaviour in generic terms – it is still rare that this perception is in line with the actual implementation status. It is important to remember that it is the actual implemented process that will make a real input on the company strategy to SOA compliance. It is then imperative to point out which processes are actually producing value by exposing the real current status of the SOA environment.

The interface design phase is a critical decision point in SOA projects. Sometimes the need to update service interface after deployment is seen as a consequence of not taking into account principles, such as reuse and composability during interface design phase. However, due to constrained time-to-market demands or increasing system complexity, service interfaces might not be as perfect as expected from the start and will need to change during system lifecycle. By delaying or refusing service interface corrections in the adequate timing, the company will only constrain its own performance during that period. The effort to keep an inaccurate interface can imply complex and workaround implementations hidden in service implementation, which will explode in complexity with each modification. A concrete cost vs. gain analysis should always support the final decision.

As argued by [Lublinsky, 2007], whenever it is required to change a service, whether its interface or implementation that might impact the service client, this situation should lead to the creation of a new service – consider each service modification as a new service. The existing service remains available in parallel with the new one, as long as the host can still handle that extra cost regarding processor and memory constraints. Dependencies of service consumers increase with the time that the service has been available.

It is recommended that each service should have a maximum of three versions simultaneously in the same runtime environment, although in practice this number is usually surpassed as discussed by [Josuttis, 2007]. This way, the client has its own time to adapt to the new interface while its old client remains operational with the previous version of that service. More, it will guarantee that a faulty new service version does not completely cut the access to that device or service. However, this approach possibly implies an explosion of versions, what introduces the need for processes to manage deprecated and remove old versions when they are no more used by previous clients, at the same time that new versions are extensively tested and made available. These processes need to be smooth enough to maintain old clients running while promoting system evolution and refinement.

A system of incentives might compel clients to remain updated to newer versions of services. By having different versions of the same logical service running at the same time, there is a need to take into account possible services dependencies, exclusive resources access or parallelism issues. A temporary solution is for the new service to be backward compatible so that an old client can continue to use the new service in the same way as before. As noted by [Brown and Ellis, 2004, Evdemon, 2005], some changes on the service interface hinder this approach, such as eliminating or renaming an operation, changing the parameters of an operation, structure of a data type or content of a message.

A service can be generic enough to be invoked by different consumers from different domains. This can also influence the constraints of new interface design – an update might be considered adequate to a consumer and at the same time have no sense for other consumer from a different domain.

The service client can also enhance its adaptability to new versions by having a thin layer that would map external data types into internal data types – the client code remains partially independent from the version of the invoked service. Another solution would be to apply a classical service broker between an old client and a new service, as discussed by [Peltz and Anagol-Subbarao, 2004]. This approach will clearly add a possible important performance cost to the system and particularly noticeable from the client side. Besides this drawback, there is also a need to modify the broker every time a new service interface is changed – one more component to be updated besides the service itself. Modifications in service interface, by default, should only impact the service provider and its consumers.

While defining service contracts as generic as possible, caution is needed since this approach can introduce complexity and hidden dependencies. For example, by using generic data types in service interface instead of typed data types, the service interface is more generic but it would not possible to detect possible miscalculations during compilation phase. These bugs can then only be detected through extensive debug or even during run-time. A common advice would be to keep the interfaces as simple and clear as possible. Versioning of data types is then an important issue to take in account in large

distributed systems.

In the same way a service-oriented system is by nature distributed, the development process is usually also distributed among different teams sometimes even across different organisations. It is then essential to agree on policies to regulate the aspects related with system versioning and management of its lifecycle.

As stated [Novakouski et al., 2012], WSDL documents should be the backbone of any service versioning strategy, being advisable to use the namespace field to differentiate services and interfaces versions as upheld by [Brown and Ellis, 2004]. [Juric et al., 2009] presents a detailed analysis of possible extensions to WSDL documents to support versioning.

A good summary of recommendations on how to tackle SOA versioning problematic is also presented in [Novakouski et al., 2012]. In the domain of SOA, versioning numerous contributions have been provided by several authors, such as [Brown and Ellis, 2004, Evdemon, 2005, Poulin, 2006, Juric and Sasa, 2010]. Regarding versioning support SOA infrastructures it is important to refer the efforts made by [Fang et al., 2007, Leitner et al., 2008]. On the standards organization sides, some initiatives have been pursued to control and manage services versioning by semantically annotating WSDL documents such as in [Sedukhin, 2005, Kopeckỳ et al., 2007].

The process of creating a consistent and clear version-control policy as part of the development effort for a service-oriented system is a still an intricate task. To ease it, there is a need to engage the developers to use a convenient software version control system sufficiently robust to accommodate all the requirements of the current development project.

Although it is important to keep in mind these constraints and set of best practices, SOA still delivers a major input when comparing with more traditional approaches. The ability to update service implementation without impact for the client due to the interface abstraction, which by nature is detached from machine specifications, programming languages or operating system can provide a fundamental contribution to the domain of industrial automation.

2.2.5 Pitfalls and Misperceptions

SOA is sometimes excessively exploited for marketing purposes and became almost a "buzzword" in the ICT business realm. This situation raised some scepticism and partially overshadowed its real meaning and purpose. As well remarked by [Erl, 2005] it is imperative to distinguish between SOA as an abstract model and an application exclusively based on web services. SOA represents a distinct architecture based on a set of distinct principles. To fully realize the potential of SOA there is a need to discuss and systematize how services are positioned and designed in the scope of the application as a whole and not simply implementing web services technology to enable communications between components. Besides that, even if a resource is exposed via a web service

that does not state the meaning of the exchanged content. Although SOA core principles are fairly simple, their concrete application can prove to be a complex task – the system simplicity is expected to emerge later when the service-oriented approach is correctly deployed. The new generation of WS-* specifications conveyed SOA to a broader audience, however just because the implementation comprises one of these, that does not consequently mean it is a SOA application. Another misperception on SOA is that it is assumed that if someone had already some experience using web services will immediately grasp the SOA vision – SOA principles are technology agnostic. Implementing SOA requires more skills than those needed to implement web services.

Due to the marketing hype of SOA many assume that just because they are exploiting SOA-based technology that it will immediate miraculously transform the enterprise into a transparent, uniform and federated organization. This goal might be achievable but it will require further investment, analysis and consensus when specifying the system requirements, architecture and implementation guidelines.

Along with the above common misperceptions, SOA principles and technology can also introduce some pitfalls to the unacquainted SOA developer. One of the most common is the endorsement of traditional distributed architectures approaches. In this scope there is a tendency to extensively adopt RPC-like service operations and synchronous MEP and inadequately divide the functional boundaries of services creating non-composable services. Also some applications include the creation of non-standard services or services dependent on proprietary technology instead of adopting available WS-* specifications. During the specification, development and deployment of a SOA application there is a vital prerequisite for broad consensus. Without it, application developers can see themselves dealing with avoidable integration issues, such as incompatible data representations, different service interfaces guidelines, dependency of conflicting software extensions, etc. Regarding system performance, each infrastructure layer has its own impact.

While the volume of message-based interactions is expected to grow there is also a strong probability of augmenting process latency if no early provisions have been set. This way there is a need to clearly determine system requirements and constraints *a priori* by testing message processing capabilities, servers availability and select an optimization solution if needed. A recurrent concern when developing SOA applications is to ensure an appropriate security level. The introduction of security procedures such as WS-Security [OASIS, 2006a] can lead to the redevelopment of the system architecture and possibly imply the choice to implement a centralized security solution.

By focusing of their own implementation technology (for example *.Net* or *Java*), a SOA developer can feel tempted to pursue of proprietary-based solution that goes against the goal of vendor neutral communications and constrain disparate applications interoperability. An effort must be conducted to avoid competing vendor alliances and proprietary extensions required to use their tools and technologies. For every SOA deployment there

is need to specify a realistic transition plan and leave your infrastructure as open as possible to cope with the latest technology developments and better handle lifecycle updates or refinements.

2.2.6 Multi-Agent systems inspiration

Part of the research conducted on the application of distributed techniques in this domain has also been supported by MAS however this approach can be considered complementary to the SOA vision as already discussed by [Huhns, 2002, Ribeiro et al., 2008c]. Even if the work presented in this document is based on SOA principles and technology it is still conceivable to be extended by a MAS application to promote added automation and proactivity to some of the tasks it supports. This way, SOA can provide the foundations upon which a more complex MAS can be built on while embedding all the added value offered by the SOA approach.

Originally from the computer science and Artificial Intelligence (AI), MAS [Wooldridge, 2002] are also becoming recognised as relevant in the context of the manufacturing world, even at the lower level aspects that include shop floor control as endorsed by [Bussmann et al., 2004]. Although having some known downsides as referred by [Wooldridge and Jennings, 1998] it is still an area in expansion. In [Monostori et al., 2006, Pěchouček and Mařík, 2008, Leitão, 2009], the authors provide reviews of existing industrial applications employing MAS-based solutions, identifying the key points, future challenges and potentials of industrial agents deployment.

Nevertheless, agents can provide industrial automation environment more autonomy, responsiveness, redundancy, distribution and robustness capacities being able to deal with incomplete information and knowledge. While most of the applications are software-based, the automation and networking applications require strict hardware constrains. The level of maturity of this technology has been identified as still particularly poor in automotive and supply chains domains. Allied to these, equipment safety, communications load, and real-time response are other issues which solutions still need to be further investigated. Particularly in this subject, there is a still big gap between academic researchers and industrial automation mainstream. The main limitations that inhibit the massive adoption of this paradigm by industry are identified:

- Narrow awareness about the true potentials of agent technology in industry.
- Reduced spread of successful industrial implementations.
- Initial over-expectations that raised some doubts and frustrations to early industrial adopters.

The combination of SOA and MAS approaches have already been attempted by some authors in order to check for its feasibility and impact. Although most results still refer to business and enterprise level applications such as the work by [Shen et al., 2007] where

web services provide the backbone for some agents to handle business transactions and therefore not targeting low cost computationally limited resources such as the ones under study. Previous work by [Lyell et al., 2003, Maamar et al., 2003, Greenwood and Calisti, 2004, Liao et al., 2004] focused on how to enable agents to request, provide or manage web services deployments.

Gradually these approaches are also reaching the industrial automation domain. The work by [Herrera et al., 2008], proposes the integration of web services and a MAS in order to build a control architecture that supports automated reconfigurability. The solution provided some support for service redundancy, composition and scheduling, at the same time it proposed a mapping between FIPA-ACL messages and WS-Addressing message properties – also known as WS-ACL. In [Mendes et al., 2009b], the authors discuss the current available solution to implement collaborative solution for industrial production systems based on a SOA infrastructure that provides the foundation for a more intelligent behaviour handle by a community of agents. Some SOA implementations also borrow some of MAS concepts to support agent-like behaviours interfaced by web services. In [Ribeiro et al., 2008a], a case study for distributed equipment diagnosis is shown where state control and the execution model were inspired by MAS MEP interactions.

Although receiving some inspiration from the MAS research territory, the principal objective of the present chapter is to investigate industrial automation device level requirements and expectations in the context of its current state-of-the-art and identify which SOA concepts and methodologies can bring some contribute to which domain segment.

2.3 Semantic Web

2.3.1 Overview

To cope with the lack of explicit semantics in the Internet domain, the Semantic Web concept was firstly delineated by Tim Berners-Lee in [Berners-Lee and Fischetti, 1999]. Semantic Web is an envisaged Internet evolution in which the meaning of online information and services is clearly characterized and linked, making it possible for the web itself understand and satisfy the requests of people and machines to exploit it, as also depicted in [Shadbolt et al., 2006, Daconta et al., 2009]. In summary, the Semantic Web is about two things: common formats for integration and combination of data drawn from diverse sources, and language for recording how the data relates to real world objects. Several organizations are actively promoting this new approach such as [W3C, 2010, SemanticWeb, 2010].

As discussed by [Lassila, 2005] with the emergence of ubiquitous and mobile computing the requirements for better interoperability foresee a higher degree of automation of many tasks that would otherwise require the end-user attention need to be increasingly addressed. Due to the high heterogeneity and dynamicity of devices and services,

new solution must be investigated to enhance discovery and device or services coalition formation without "a human in the loop".

Based on previous premises, the concept of semantic web services was born as complement to traditional web services technologies, as presented in [McIlraith et al., 2001, Payne and Lassila, 2004]. In order to promote dynamic, scalable and cost-effective infrastructure for electronic transactions, the Semantic Web Services research community [W3C, 2008,SWSI, 2009,OASIS, 2010] is pushing forward by enriching web services with machine-processable semantics. Since web services technology is the most deployed technology to implement service-oriented applications, the SOA paradigm can receive here an important input.

2.3.2 Ontology

When dealing with semantic web aspects, the concept of ontology is always an essential cornerstone. The definition of the ontology concept still generates some discussions and controversy among the AI community. These definitions mostly vary in accordance with its domain of application and author background as depicted in [Gruber, 1993, Uschold and Gruninger, 1996, Gómez-Pérez et al., 2002, Happel and Seedorf, 2006, Staab, 2009]. In the current context, an ontology defines the concepts and their relationships in the scope of a particular domain of interest, providing a vocabulary for that domain as well as a computerized specification of the meaning of the terms used in that vocabulary. An ontology is intended to be used during knowledge reasoning about the domain. Ontologies range from taxonomies and classifications, database schemas, to fully axiomatized theories. In recent years, ontologies have been adopted in many business and scientific communities as a way to share, reuse and process a domain knowledge. Ontologies are now crucial to many applications such as scientific knowledge portals, information management and integration systems, AI, biomedical informatics and electronic commerce.

In this area, OWL [McGuinness et al., 2004] is becoming a worldwide knowledge representation language for the web and supports different levels of semantics. OWL is based on earlier languages OIL and DAML+OIL. OWL is acknowledged as the most frequent choice for developing web-based ontologies. It has a strong community support, as well as several available tools to support user development. An OWL ontology may include descriptions of classes, along with their related properties and instances. OWL is designed for use by applications that need to process the content of information instead of just presenting information to humans.

Ontology editors are tools designed to assist the user during the creation and manipulation of an ontology. Several tools are currently available: Knoodl [Revelytix, 2012], TopBraid Composer [TopQuadrant, 2012], Protégé [Gennari et al., 2003] and WSMO Studio [Dimitrov et al., 2007]. The large majority of the semantic web research community elects Protégé as their tool of preference when regarding ontology engineering, although WSMO Studio is also slowly gaining popularity. Protégé is a free, open-source platform

that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontologies. At its core, Protégé implements a rich set of knowledge-modelling structures and functions that support the creation, visualization, and manipulation of ontologies in various representation formats. It can be also customized to provide domain-friendly support for creating knowledge models and entering data. Further, Protégé can be extended due to its plug-in architecture and an Application Programming Interface (API) for building knowledge-based tools and applications. Following latest versions tendencies and success, the latest version of Protégé is now built on top of an OWL API [Manchester University, 2012], providing more flexible development and a straightforward migration path for OWL-based applications.

A reasoner is also a central resource when dealing with ontology-based reasoning. A reasoner is a software component able to infer logical consequences from a set of asserted facts or axioms to check whether or not all of the statements and definitions in the ontology are mutually consistent and recognise which concepts fit under which definitions – ontological classification. There is also an available set of reasoners available such as RACER [Haarslev and Müller, 2001], FaCT++ [Tsarkov and Horrocks, 2006] or Pellet [Sirin et al., 2007]. As described by [Guo et al., 2011], the performance of a reasoner is mainly affected by the quality of inputs, optimization techniques and feasibility for customization.

2.3.3 Key Specifications

Several recent specifications on the subject are already available, such as Semantic Markup for Web Services (OWL-S) [Martin et al., 2004] and WSMO [Roman et al., 2005] regarding web services ontologies, and SAWSDL [Kopeckỳ et al., 2007] for semantic annotation of web services.

OWL-S (Semantic Markup for Web Services) is an ontology of services that also pursues the automation of web service tasks including automated service discovery, execution, interoperation, composition and execution monitoring. OWL-S supplies web service providers with a core set of markup language constructs for describing the properties and capabilities of their web services in unambiguous, computer-processable form. The OWL-S ontology has three main parts: the service profile for advertising and discovering services; the process model, which gives a detailed description of a service's operation; and the grounding, which provides details on how to interoperate with a service. One of the aspects sometimes criticised of OWL-S refers to its lack of methodologies to support faults in a more straightforward manner as depicted by [Martin et al., 2007]. The development of the OWL-S specification took mainly in account the following features:

- Automatic Discovery: service discovery can be supported by computer-interpretable semantic markup at the service website, and a service registry or ontology-enhanced search engine could be used to locate the services automatically

- Automatic Invocation: OWL-S markup provides a declarative, computer-interpretable API that includes the semantics of the arguments to be specified when executing these calls, and the semantics of that is returned in messages when the services succeed or fail.

- Automatic composition and interoperation: the information necessary to select and compose services will be available at the service websites. Software can be written to manipulate these representations, together with a specification of the objectives of the task, to achieve the task automatically. To support this, OWL-S provides declarative specifications of the prerequisites and consequences of application of individual services, and a language for describing service compositions and data flow interactions.

In a similar way, the Web Service Modelling Ontology (WSMO) provides a conceptual framework and a formal language for semantically describing all relevant aspects of web services in order to ease discovery, composition and invocation of electronic services. Having the Web Service Modelling Framework (WSMF) as support base, this one is refined and extended through a formal ontology and a Web Service Modelling Language (WSML). WSMF defines four different main elements for describing semantic web services: *Ontologies* to provide the terminology used by other elements; *Goals* to define the intentions that should be solved by a service; *Descriptions* that define various aspects of the service; and *Mediator* to resolve interoperability problems. WSMO is based on the next design principles:

- Web Compliance: WSMO inherits the concept of URI for unique identification of resources and namespaces for denoting consistent information spaces.

- Ontology-Based: all resource descriptions and all data interchanged during service usage are based on ontologies.

- Strict Decoupling: decoupling denotes that WSMO resources are defined in isolation, meaning that each resource is specified independently without regard to possible usage or interactions with other resources.

- Mediation: mediation addresses the handling of heterogeneities that are always present in open environments. Heterogeneity can occur in terms of data, underlying ontology, protocol or process.

- Role Separation: service consumers exist in specific contexts which will not be the same as for available web services. The underlying epistemology of WSMO differentiates between the desires of users or clients and available services.

- Description versus Implementation: WSMO differentiates between the descriptions of Semantic Web services elements (description) and executable technologies (implementation).

- Execution Semantics: In order to check a WSMO specification, the formal execution semantics of reference implementations like WSMX [Haller et al., 2005] to provide the technical realization of WSMO.

- Service versus Web service: A Web service is a computational entity which is able through an invocation to achieve a users goal. A service in contrast is the actual value provided by this invocation.

As investigated by [Lara et al., 2004] most of the description elements defined in OWL-S can be modelled in WSMO, although some aspects, such as the specification of the service orchestration and the WSDL grounding are more detailed in OWL-S. A more complete comparison of each aspect of these two specifications is also presented by these authors.

The Semantic Annotations for WSDL and XML Schema (SAWSDL) specification based on the original WSDL-S proposal defines how to add semantic information into WSDL documents. These semantic annotations are used to automate service discovery, composition, mediation, and monitoring. Semantic annotations define the meaning of the inputs, outputs, preconditions and effects of the operations described in a service interface. These annotations reference concepts in an ontology.

When comparing SAWSDL with OWL-S and WSMO it is possible to extract some important advantages:

- Possibility to describe both the semantics and operation level details in a single WSDL file – a language that the developer community is familiar with.

- It consists on an agnostic approach to ontology representation languages. It allows developers to annotate their web services with their choice of ontology language (such as UML, OWL or WSML) unlike in OWL-S. This feature eases the reuse of existing ontologies, independently of their modelling language.

- It does not duplicate descriptions already defined in WSDL (ex. input and outputs).

These two approaches can still be combined as discussed by [Paolucci et al., 2007] when pursuing to automate OWL-S Grounding based on SAWSDL annotations or through a set of recommendations as in [Martin et al., 2007].

3

A Survey on SOA for Industrial Automation

3.1 Introduction

As introduced in section 2.2, *SOA establishes an architectural model that aims to enhance the efficiency, agility, and productivity of an enterprise by positioning services as the primary means through which solution logic is represented in support of the realization of strategic goals associated with service-oriented computing* [Erl, 2005]. Originally emerged from ICT business world, SOA paradigm is expanding its range of application into several distinctive environments.

Industrial automation is increasingly interested on adopting SOA as a unifying approach with several advantages over traditional automation since the first SIRENA project implementation success [Jammes and Smit, 2005a]. Contemporary automation systems are always planned to be predictable, reliable and efficient however history proves that unexpected behaviour arises, equipment fails or degrades while changes on requirements involve a continuous demand for reengineering interventions. Uncertainty must be always taken into account and reengineering processes need to be sufficiently agile to cope with it. Even though other initiatives already explored the application of SOA principles into the domain of industrial production, it is still imperative to focus over the needs at device level and explore which features, tools and services employing SOA concepts and methodology can turn out to be fundamental and provide an important contribute to the domain.

Agility is a fundamental requirement for modern industrial automation companies in order to face challenges induced by the globalization, environmental and working conditions regulations, improved standards for quality and fast technological mutation.

A modern agile organization requires a smooth integration and alignment between high level processes, such as complex supply chain management, and device level automation behavior. If a lower level environment cannot accomplish the desired agility goals, the overall system will be incapable of delivering the expected agile performance.

By embedding more intelligence is each device, the overall system agility is expected to improve by having more autonomous, intelligent and self-contained devices. Also, the level of abstraction and complexity of the exchanged information will rise and devices low-level intricacies will be circumvented. Devices easier to setup, manage, monitor and diagnose are a key-factor during reengineering or down-time phases by saving a considerable amount of integration time that subsequently conditions production capacity. The connection between shop-floor devices and the enterprises services is fundamental to create more sophisticated high level services and to support more reliable decision making and feedback.

3.2 SOA input in Industrial Automation Device level

As in other paradigms, SOA defines that the logic required to solve a large problem can be better constructed, carried out, and managed if it is decomposed into a collection of smaller, related pieces, although the key is the manner in which it achieves separation, as discussed in [Erl, 2005]. As society, individual companies are service-oriented and collectively, their businesses comprise a community. By letting business self-govern their individual services, they evolve relatively independently from each other, avoiding tight connections. Still, some baseline conventions need to be followed. By standardizing key aspects to the benefit of the consumer, it can choose what services best suits its needs and exploit them in an open and uniform way.

Alongside the widely deployment of service-oriented applications at business ICT level, technology continues to evolve at an increasingly pace. The continuous convergence between computing and networking areas, as the advances in semi-conductor and transmission technology, allows new approaches to communication between systems and devices, in particular, embedded devices. At crescent rhythm, Internet-based technology is emerging as the basic carrier for interconnecting devices in the most disparate application domains. As discussed by [Jammes and Smit, 2005b], this tendency is also visible in industrial automation as a result of several converging progresses:

- Availability of low cost, high-performance, low-power electronic components that support an embedding unprecedented computing power into ever tinier apparatuses.
- Ethernet networks are becoming widely accepted as the medium of choice for device networking. On top of these networks, Internet protocols of the TCP/IP family are becoming the standard vehicle for exchanging information between connected devices.

- The rise of data interchange mechanisms based on XML enabled high-level interaction standards.
- The advent of the web services technology for interconnecting heterogeneous applications on the basis of a lightweight communications infrastructure paves the way for an universal, platform- and language-neutral connectivity.

It is clear that SOA is already having a significant impact in many branches of technology, not exclusively in original ICT sector, but also in other areas where these methodologies can be adapted. One of the most promising approaches refers to its application at device level where the usage of high-level service-based communications infrastructure can contribute to more advanced solutions.

Web services-based technology is currently the most adopted solution to implement SOA applications. The traditional impression about web services simply refers to a static software system designed to support interoperable machine-to-machine interaction over a network, being usually just a set of methods that can be accessed over a network and executed on a remote system hosting the requested services as presented in [Cerami and St Laurent, 2002, Haas and Brown, 2004]. The application of web services in several distinct areas promoted the creation of more specific specifications covering different subjects such as discovery, security, addressing, reliable messaging, etc. These specifications are often combined to create profiles that adapt to a particular domain of application.

Being commonly supported by open technology standards, web services ease the interoperability, integration and reuse of the application components. This way it is possible to deliver a direct and seamless communication between devices and ERP levels. In this scenario, a device embedding its own services can directly supply necessary data, physical placement, number of previous identical faults, etc.

Taking into account SOA premises and industrial automation device level requirements, the subjects most interesting to be investigated and transferred to this new domain of application are as in [Jammes and Smit, 2005b]:

- The service design follows an outside-in approach, i.e. the interface of the service is defined by focusing on how that service can fit as an atomic task in a larger process;
- Services are possible to be composed into higher-level services increasing its complexity and abstraction;
- Communications are loosely coupled and of an asynchronous nature;
- Services abstract heterogeneous hardware and software platforms from different providers. As each service encapsulates its own complexity, scalability, manageability and maintenance become built-in features.
- Application components collaborate by sharing information and resources, in a peer-to-peer manner, in any configuration layout; i.e. from totally distributed to strictly hierarchy;

- It enables genericity and reuse;
- Ability to negotiate their properties, such as QoS, security level, performance constrains, etc.

In a service-oriented production system, all elements expose their abilities in the form of services. These elements are able interact to share other types of data besides common service invocations. These interactions may include discovery, metadata exchange, eventing or even "heartbeat" features. Besides the need to easily deploy devices and services, there is an emergent need to assist, and even automate the process of identification, setup and management of these resources in an agile manner along the system lifecycle.

Nowadays the large majority of industrial automation installations are still not SOA-compliant and that is the reason why some manufacturers are interested in new integration approaches. Since the original SIRENA [Jammes and Smit, 2005a] implementation, several other works have exploited SOA concepts and technology to ease the integration of their equipment in a service-oriented infrastructure. In fact, the most common applications in the latest deployments of SOA in the industrial domain device level address the abstraction of equipment instances and enterprise integration. In [Gilart-Iglesias et al., 2006b] the authors envisioned an "Industrial Machine as a Service" (IMaaS) to support the visualization of manufacturing devices from a functional perspective. The ultimate goal of this approach was to raise the abstraction level of manufacturing devices from the lower levels of production to the enterprise level business model. In [Feldhorst et al., 2009], a traditional Programmable Logic Controller (PLC) system was substituted by a few embedded industrial PCs and a layer of device facades was implemented to expose the current system in the form of reusable services over Ethernet. These device facades implement a hardware abstraction layer that decouples higher-level control systems from low-level implementation details. This solution warrants low-level real-time requirements, as long as a suitable level of services granularity is followed – services that require critical time constraints were kept out of the scope of the service interface. Also in respect to integrate wireless sensor networks into other existing IP-based networks, similar SOA approaches have been followed as in [Sleman and Moeller, 2008]. This work adopted DPWS to create a gateway suitable for embedded devices in home and industrial automation due its intrinsically nature in terms of interoperability, automatic networking and services peer-to-peer discovery.

A SOA-capable device gains in interoperability, autonomy and openness to be employed as an important brick over which complete infrastructure can be based on. Not only providing a common background communication technology and middleware, but also providing embedded added value over which more complex applications can be deployed faster and still remain agile enough to cope with mutable system requirements. Based on previous requirements and investigations, the foreseen SOA input can be split into different sub-domains of the industrial automation device level as presented in Fig. 3.1.

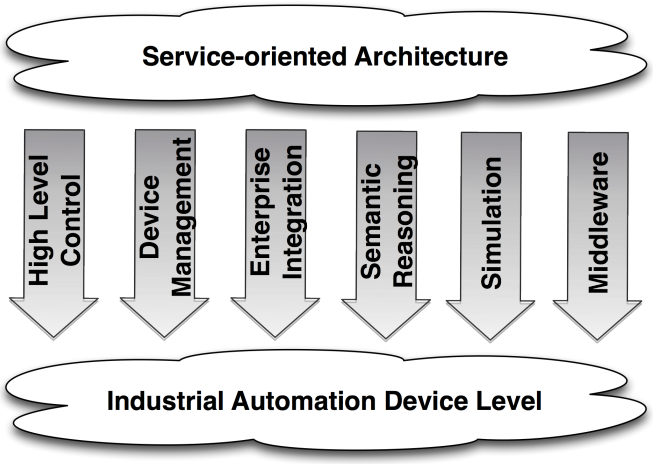


Figure 3.1: SOA contributions to industrial automation device level

3.2.1 High Level Control

To promote system agility, the design and development tools should straightforwardly and intuitively allow the integrator to build his application easily and faster than older approaches with, at least, the same level of robustness and performance – this is the key aspect to a wider adoption of SOA on any domain of application as depicted by [Bloomberg and Schmelzer, 2006]. The aspect of control within SOA approaches is mostly related to the process of making several services work together to create added value in the form of a more complex process. The interactions between services are mostly defined as a sequence of procedures, most of the time owned by different distributed entities, which needs to be followed to accomplish the expected goal.

3.2.1.1 Real-time concerns

The recurrent requirement for real-time due to the strict performance and safety processes in industrial automation domain is amongst the most common critics when referring the employment of SOA approaches, particularly when addressing control aspects. Although some work has been conducted to introduce real-time capabilities to SOA application, the majority of the existing SOA deployments in this domain still consist of test-benches with no genuine concerns on real-time constraints.

Some works address the QoS integration in web services but with no concrete technical solution to solve the level of real-time required at device level in industrial automation. The work conducted by [Tian et al., 2003] enabled QoS integration in web services, but also the selection of appropriate services based on QoS requirements in terms of server and network performance. The authors in [Tsai et al., 2006c] present the Real-Time SOA (RTSOA) framework but it simply consists of a set of guidelines and algorithms mostly focusing again on general QoS aspects. As pointed out by [Menasce,

2002], one of the biggest problems on using web services is the existence of XML parsers on both communications ends.

Regarding the work pursued to support real-time web services solutions in the industrial automation there are some recent noteworthy results that should be explored. In [Delamer and Lastra, 2006a] the authors propose an extension of the CAMX SOAP/XML framework with QoS support where new XML messages are described for regulating the interactions among middleware instances. This application exploits a services differentiation solution for IP networks based on a set of aggregated data flows to enable fast and reliable communication between peers. Other experiments by [Moritz et al., 2008] have also shown that it is possible to run complex web services on top of a real-time Operating System (OS) with bounded response times. In [Pohl et al., 2008], a SOA test-bench based on a DPWS implementation supported by Java Real-Time System (RTS) is assessed, however the evaluation platform relies on PC-level hardware and does not really address the low-resources devices commonly available at industrial automation device level.

As referred by [Mathes et al., 2008], hardware and software platforms at industrial automation device level are less powerful than those from upper ICT layers and automation engineers have no real experience on developing and deploying web services. Since industrial processes typically have time restrictions, particular attention has to be paid to the description of such constraints within the web service implementation to provide a mean to support the timely execution of these in the current infrastructure. The same author in [Mathes et al., 2009b] in the scope of Time-Constrained Services (TiCS) development also introduced the SOAP4PLC engine that extends an IEC61131-3 compliant programming system with an interface for exporting PLC functions. For each exported PLC function, an according web service is deployed automatically and a WSDL document is generated. In a similar solution tackling industrial PCs instead of PLCs (SOAP4IPC) [Mathes et al., 2009a], it employed a SOAP engine that allows web service invocations within predefined time constraints deployed on industrial PCs. In the scope of RI-MACS project, the work by [Cucinotta et al., 2009] proposed a solution for real-time traffic management by extending the WS-Agreement protocol to support real-time and QoS features. However, it is also stated by the authors that increasing the computation power on which software is running is not enough, in general, for securing precise real-time requirements.

The work depicted in [Jammes et al., 2012] trailed in the scope of the ongoing FP7 AESOP collaborative project tries to answer some common critical questions posed when employing SOA concepts and technologies to large scale process monitoring and control systems. As further detailed in [Jammes, 2011], although DPWS can provide a standard solution for implementing SOA at device level as confirmed by previous results, the achieved performance is still sometimes not sufficient. This situation is especially critical for low-level process control. To solve this problem the chosen solution is to couple DPWS and Efficient XML Interchange (EXI) [Schneider and Kamiya, 2011], the most promising Binary-XML specification coming from W3C that addresses both the XML size

and processing performance issues. This approach is expected to reduce the amount of device resources in order to encode and decode SOAP/XML frames.

Although real-time web services solutions are still rare and not widely deployed or even recognized, **it is evident that it will be soon possible to warrant real-time performances that compete against current conventional industrial automation technology**. This position is even more trustworthy due to the fact that hardware resources in terms of Central Processing Unit (CPU) and memory continues to increase for the same range of prices. The research challenge is now **how to transfer and adapt SOA approach into the scope of systems integrators without compromising the acquired know-how or involve extensive training**.

3.2.1.2 SOA-based control approaches

Currently within this area of SOA-based control there are mainly two approaches: orchestration and choreography as explained by [Peltz, 2003b]. These approaches are defined in terms of global behaviour, i.e. how singular services are organised to work together.

3.2.1.2.1 Orchestration

In the orchestration approach, a central node controls a workflow that interacts with a set of services following a predetermined logic during the execution of the according complex process. The workflow logic involved in orchestration consists in several rules, conditions and events, i.e. it specifies how different entities should interoperate with the central node in order to carry out a predefined task. Here, the process logic is centralized yet still extensible and composable, being at the same time a way to abstract a process in a single service. A heterarchical approach is also possible by having several orchestrators at different levels of composition, i.e. one of the partners of the orchestration can be itself another orchestrator that encapsulates and orchestrates other partners in a hierarchical way, as exemplified in Fig. 3.2. Since each orchestrator has its own process logic, it is possible to imagine an orchestrator that some time during its process execution needs to invoke a service provided by another orchestrator (at the same composition level or any other), and vice-versa.

In a broader spectrum of application some authors such as [Peltz, 2003a, Dustdar and Schreiner, 2005, ter Beek et al., 2007] already surveyed the current web services orchestration domain. In this context there is an evident need for a set of programming constructs to describe workflow, correlate services interactions and build more complex composed services. Other common requirements include the asynchronous support to ensure process reliability, strong transactional semantics and exception handling for both internal and external errors.

Although there is an available assortment of standards for web services orchestration, the most employed are consistently Web Services Business Process Execution Language (BPEL) (also known as WS-BPEL, BPEL4WS or simply BPEL) [OASIS, 2007] and

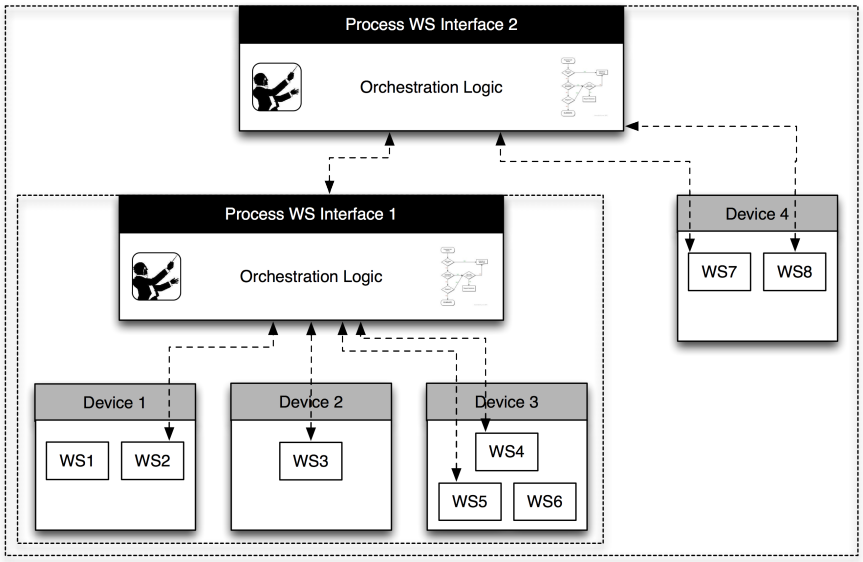


Figure 3.2: Heterarchical orchestration example

Business Process Modelling Language (BPML) [Thiagarajan et al., 2002], although recently this last one is becoming deprecated in favour of Business Process Modelling and Notation (BPMN) [White, 2004]. Developed in the scope of the necessity to specify Business Process Models (BPM), there are still some constrains when merging both technical and business aspects of a process. It should be avoided to include technical functionality within a BPM processes. These technical process specificities should be delegated to a BPEL model. Nevertheless, there are still some ambiguity and confusion when sharing BPM and converting BPMN into executable platforms. BPMN and BPEL have distinctive usage areas but certainly also overlapping functionality. BPEL is currently the primary specification related to services orchestration for the most common SOA applications, particularly in business or enterprise level deployments. BPEL structures the workflow logic with predefined primitive activities. Basic activities (*receive*, *invoke*, *reply*, *throw*, and *wait*) correspond to fundamental workflow action that can be assembled using the structured activities (*sequence*, *switch*, *while*, *flow*, *pick*).

As stated by [Jammes et al., 2005b], these orchestration solutions are made to run on top of an application server and denote too big binary footprints for small devices with low resources. For this reason, a lightweight orchestration engine is required but no such implementation was available at the time. In the scope of FP7 SOCRADES project, the author of [Mendes et al., 2009a] explored the application of Petri Net-based orchestration. The graphical and logical approach promised an easier adoption among the traditional automation community that is mostly used to IEC61131-3 languages. The continuation of this work of [Mendes et al., 2012] discussed the lower complexity of these nets in terms of implementation and the hypothetical better performance for device integration,

although not following any recognised orchestration model for web services. Due to its graphical and mathematical notation the analysis, validation and simulation of the system is immediately possible during design before the deployment phase.

The work by [Puttonen et al., 2008, Villaseñor Herrera et al., 2011] investigated the use of a MAS to support the dynamic orchestration of web services hosted by devices in manufacturing systems. A BPEL engine was chosen to execute orchestration processes, even if some limitations were detected when there was a need to match different web services in dynamic workflows – semantic incompatibility. It should be also noted that these agents were not running on devices, but only ensure the orchestration of the services hosted on these. In order to tackle this need, the work by [Loskyll et al., 2011] promoted a semantic service discovery and orchestration system supported by current semantic web approaches to support the creation of adaptive production processes A different approach was followed by [Theorin et al., 2012] that integrated DPWS and a *Grafchart* engine implementation to support the design of processes using Graphcharts that invoked and handled service invocations. This approach can be seen as a trade-off between both domains: endorsing SOA principles and technology and at the same time using a language known by automation staff.

Table 3.1 summarises the advantages and drawbacks of employing an orchestrator approach.

3.2.1.2.2 Choreography

Choreography defines a same-level collaboration behaviour between distributed participants. The goal is to set up an organized collaboration between different distributed services without any other entity controlling the collaboration logic, as discussed by [Peltz, 2003b]. Choreographies allow the definition of patterns to execute a particular task. A choreography schema assumes that there is no owner of the global collaboration logic, contrary to orchestration where the process execution is controlled and incorporated in a single element.

In order to expose a certain choreography, each service must know its own role within

Advantages	**Drawbacks**
+ Process workflow logic is encapsulated at a single point, which makes it easy to modify without impacting other process entities. + Can be applied recursively to support a federated solution +Abstracts the interactions between the orchestration node and the individual process partners, exposing itself as a single service. + Preserves the autonomy of each of the process atomic services	- no horizontal interaction – by definition, it is a totally hierarchical approach, where from the bottom to top there is always a node that abstracts and coordinates the ones below it. - Pulls process control decision out of the devices. - The orchestrator is a single point of failure.

Table 3.1: Orchestration advantages vs. drawbacks

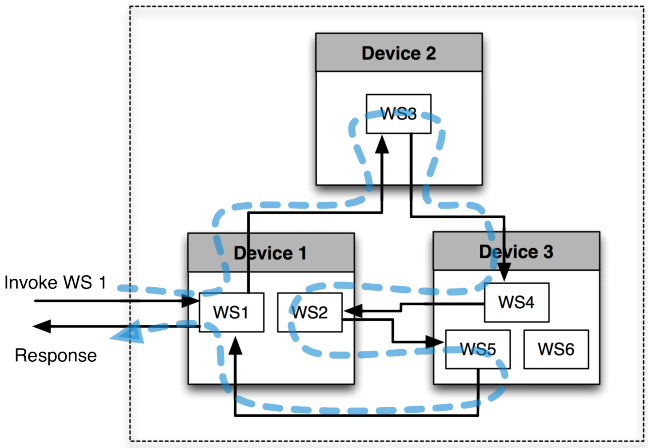


Figure 3.3: Choreography example

the current process; i.e. what the service supports and how to react or proactively execute in a particular context. These services are also referred as participants. Each possible contact between two roles in a choreography is identified as a relationship. Multiple participants can assume different roles and have different relationships. The moulds of this same relationship, i.e. the MEP between two roles, are expressed by channels. Channel information can also be transmitted through a message between services, so that a service can determine how it can interact with a particular service. Furthermore, the message exchange logic is encapsulated within an interaction – the fundamental choreography building block. Choreography, in the same way as orchestration, can have different levels of abstraction. A choreography can be composed by other choreographies in a block-based composition in order to create a more complex collaboration behaviours, as the example in Fig. 3.3.

In matters of web services coordination the focus is mostly concentrated on orchestration-based solutions, in particularly BPEL. Orchestration mostly endorses notations that use the common view approach, while choreography strongly relates with an individual view approach.

The creation of Web Services Choreography Interface (WSCI) [Arkin et al., 2002] defined an XML-based interface description language that worked in conjunction with WSDL. The original goal was to employ web services to support the creation to business processes that mirror society dynamic and ever-changing business needs. As presented by [Brogi et al., 2004], WSCI can even be formalized using a process algebra approach and this can enhance formalization and validation aspects. The development done for WSCI led to the creation of its substitute specification: Web Services Choreography Description Language (WS-CDL) [W3C, 2005]. WS-CDL describes peer-to-peer collaborations of participants by defining from a holistic point of view their common and complementary observable behaviour and where ordered message exchanges result in accomplishing a

common business goal. While WS-CDL provides reactive rules that are used by each participant to compute the state of the choreography and infer which message exchange will occur after, BPEL specifies active rules that are executed to infer what to do next and executes the corresponding task. In [Barros et al., 2005], the authors enunciate some of the issues that still need to be further improved in this specification: no strict separation between semantic and syntactic aspects; and since service choreographies are more a design than an implementation item, a modelling notation for service choreographies would be more useful than simply an XML syntax. The same authors propose some topics for further investigation: documentation of a library of service choreography patterns, definition of a service choreography meta-model, and define concrete syntaxes for service choreography interchange.

As discussed by [Leavitt, 2004], the original goal of using web services was to unlock a new generation of e-commerce applications. When discussing choreography, the author supports that a well-defined choreography guarantees conformance across application domains and businesses reduce their time to market. As depicted by [Zur Muehlen et al., 2005] choreography aims at the coordination between distributed peers that use web services to expose their available operations. The coordination of long-running interactions is necessary in almost any organizational interaction.

In the context of industrial automation, besides the initial work done by [Delamer and Lastra, 2006b] the majority of the authors prefer to research more orchestration-oriented solutions, which are somehow closer to current practices when modelling an industrial process: a "master" element that coordinates its "slaves" to perform a predefined task.

A summary of choreography advantages and drawbacks is presented in Table 3.2.

3.2.1.2.3 Discussion

Both previous approaches for SOA-based control have different goals depending on the application in study. While choreography denotes a more self-organization approach, orchestration still endorses a more hierarchical approach in line with traditional industrial automation control approaches, although still flexible enough to cope with heterarchical topologies. An orchestration can also be regarded as specific application of choreography. In fact, WS-CDL and BPEL specifications mutually overlap in some features due to its different standards organizations origins, respectively W3C and OASIS.

Advantages	**Drawbacks**
+ Allows the definition of a completely distributed control approach. + Supports distributed complexity. + Services more autonomous (embedded intelligent control decision). + Possible standardisation of devices interaction for some simple common applications	- Necessary to divide and distribute the workflow logic to all involved devices, (although less locally complex). - Possible network traffic boost when a large number of services are connected and active. - Difficult to scale to large and complex applications.

Table 3.2: Choreography advantages vs. drawbacks

For simple use cases, choreography can increase autonomy by embedding some level of intelligence on application resources (device and services) and enhancing loose-coupled interactions between them. As example, this approach can be adopted by an intelligent system of conveyors. By embedding intelligence to each module (conveyors, transfers, lifts, etc.), it can automatically detect and recognize its "neighbours" and, together, define which is the best path for a particular pallet to be conveyed, avoiding congestion paths and faulty equipment, in a similar way as packet transport algorithms are used in telecommunications domain. Here, there is no central unit to coordinate the transport system as a whole. The global system adapts to the current situation based on local information available from each module and the interactions between them – self-organization.

A traditional example of an orchestration application would consist of a machine composed by several layers of software abstraction. Each layer is composed by elements that abstract that level capabilities to the upper level, and so on, from I/O values to complete machine/workstation abstraction and even higher. This way, the above layer does not need to deal with previous layer intricacies being a simple consumer of the clean and self-describing services provided by the level below. Here, orchestration provides enhancements on services composition and hierarchical coordination.

In a complex environment with several complex machines internally controlled by an orchestration approach, choreography can be used to manage the interactions between them. This would consist of a mixed approach, by having orchestration controlling machine building blocks and exposing the complete machine as a service and then having this same services autonomously interacting to achieve a goal in choreography-like manner. It is evident that neither one of these approaches will arise as the final solution for SOA-based control, although orchestration approaches are getting a lot of attention in this moment. The approach to choose is always conditioned by the requirements for each application.

Service-based interactions should be kept within the scope of the device access and management, and application services should be used to interface more high-level process-oriented tasks, if no real-time web services solution is available in that context. These services should act as an endpoint to execute more abstract processes where the service interface clearly states its function and output instead of a collection of illegible I/O *read* and *write* operations.

3.2.2 Device Management

The term device management started to gain some importance in the domain of network and mobile communication devices. Several different definitions can be retrieved for device management varying accordingly with the domain of expertise of the source. In this context, device management refers to the process of acquiring, configuring, deploying, securing, maintaining and decommissioning devices in order to deliver specific business

and end user benefits.

Firstly delineated in the late 1980s, the OSI/ISO Network Management Model [Yemini, 1993] was proposed as a set of protocols to tackle first networks management issues but at the end it became mostly known as a reference framework that stated the different spheres of network management domain. This model led to the creation of the Common Management Information Protocol (CMIP) [Warrier and Besaw, 1989], which later lost to its competitor Simple Network Management Protocol (SNMP) in terms of broad industry adoption although by default it supported more features than the latter. This management model is also known by the FCAPS acronym which emerges from the first letter of each one of the areas of functionality described in the model, as in the following list:

- *Fault*: recognize, isolate, correct and log device errors to improve its easier and faster resolution.
- *Configuration*: retrieve, set, update and monitor devices configuration parameters.
- *Accounting/Administration*: collect usage information for ranges of device users.
- *Performance*: retrieve device and network statistics to better evaluate current system performance and prepare for future improvements.
- *Security*: control and manage the access to critical resources of system.

Several protocols, specifications and methodologies were developed along the time to deal with these areas of functionality as a whole or only tackling some individually. Although the domains of applications constrained the focus of the management applications, some protocols remained generic enough to adapt to several domains.

SNMP [Feit, 1993] is still the most widely used management protocol in production. It is a technical solid solution low on resources that runs fast. Still, it cannot be seen as a perfect solution for more contemporary demands. Although initial critical issues regarding a bad security model were partially solved in the third version [Stallings, 1998], this last one was never widely adopted. Furthermore, emergent issues arose concerning the large amount of proprietary Management Information Bases (MIBs), mostly overlapping but still incompatible between them and most of the times not entirely documented. The setup phase turns to be complex either due to poor documentation or application of proprietary Object Unique Identifiers (OID) layouts – difficult to find and interpret unknown or property data. Regarding data types, it is also delicate to describe complex data structures.

As depicted in [Schoenwaelder, 2003], some survey results from network operators confirmed that SNMP was mainly used for network monitoring and not to configure network equipment as it was initially supposed to. It became as well evident that SNMP could not handle sophisticated management since its ineffectiveness limited its domains

of application to simple monitoring. Therefore, alternative approaches were required to accomplish management goals such as efficiency in information retrieval, transaction support, and security together with reduced development and operational costs.

Several authors such as [Leppinen et al., 1997,Haggerty and Seetharaman, 1998,Pavon et al., 1998] amoung others promoted the use of CORBA technology to enable network management, however with the decline of CORBA applications these solutions were also gradually abandoned.

Receiving the input of XML-based technology, Netconf Configuration Protocol [Enns, 2006] was created trying to take into account the needs of both network operators and equipment vendors. Netconf lies over a RPC layer on top of the transport layer and applies XML data encoding both for configuration data and protocol messages. The base protocol, including basic primitives to get, edit and delete data, locking and session management can be extended with further features such as eventing and retrieval of schema definitions. This protocol can also employ SOAP as transport mapping [Goddard, 2006].

Following this trend, Sun Microsystems put together its own solution regarding resource management: Java Management Extensions (JMX) Technology [Perry and Denn, 2002]. It also offers a simple and common way of managing resources such as applications, devices and services. A resource is represented by an object called Managed Bean (MBean), that can be dynamically loaded and instantiated, being then possible to monitor and manage it via the JMX API. Although being exclusive to Java-based systems, this architecture can be envisaged to be applied as a wrapper solution for legacy systems.

In a scope more close to mobile devices and applications, Open Mobile Device Management (OMA DM) [Open Mobile Alliance, 2008] is also trying to unify management approaches in this domain by promoting a single generic solution to manage different devices populations. Due to mobile devices nature, this specification is intended for small footprint device with possible constrained connectivity and guaranteeing tight levels of security on access. Although based on an XML-based format known as SyncML, it mostly relies on mobile protocols such as GSM, CDMA, Bluetooth, SMS or WAP.

In the domain of embedded automation some lightweight protocols try to address these management needs but still in a very small scale and relying on low level communication solutions mostly focused on ensuring real-time monitoring. As for Network Management (NMT) set of protocols in CANopen [Boterenbrood, 2000], the big goal is to execute device commands and detect fault situations. The *Heartbeat* protocol that is used to monitor network devices by verifying if they send a binary message stating that they are "alive" with a predetermined time limit is also included in this set of protocols.

Web-Based Enterprise Management (WBEM) introduced by [Thompson, 1998] and now developed and maintained by Distributed Management Task Force (DMTF), is a collection of management and standard technologies developed to unify the management process of distributed computing environments and it was envisaged to replace most of SNMP functionalities. WBEM is based on open standards as Common Information Model (CIM) infrastructure and schema, CIM-XML, CIM operations over HTTP and

WS-Management. In summary, it envisages easy and open access and discovery of management elements. Several implementations of WBEM are available in open source such as OpenWBEM [OpenWBEM, 2006], OpenPegasus [The Open Group, 2008], WBEM-Services [Sun Microsystems, 2004], Purgos [SofTulz, 2009], PyWBEM [Potter and Whiteley, 2008], SBLIM [IBM, 2008], and WMI [Microsoft, 2009].

Coming also from DMTF, CIM [DMTF, 2003] is an open standard that defines how systems, networks, applications and services can be represented as a common set of objects and relationships between them. This specification is intended to support consistent management and exchange of semantically rich elements, independent of their manufacturer or provider, throughout the network. CIM also provides the means to actively control and manage these same elements. CIM comprises CIM Schema and CIM Infrastructure Specification, both used in WBEM architecture. The CIM Infrastructure Specification defines an UML-based architecture and concepts, including a language by which the CIM Schema and its extensions are defined, and a method for mapping CIM to other information models, such as SNMP. The CIM Schema defines a collection of objects and relationships between these, representing a standard base for the elements to be managed. The CIM Schema tries to cover the majority of traditional elements in an ICT environment, such as computer systems, operating systems, networks, services and storage. Due to product and vendor specificity, CIM Schema is possible to be extended to allow them to represent their specific features seamlessly together with the common base foundation of CIM Schema specification. The capability to use generic information defined by CIM models covering a broad range of generic applications that can interoperate with each other was the main reason why original Object Linking and Embedding (OLE) for Process Control (OPC) specifications were a success from the start. Since the main objective of OPC standards is to ensure the consistent exchange of data between all OPC-enabled automation components and the control system, the management aspect is also included. This way, management communication is also dealt following previous premises: components data are defined using known CIM models and data is set or retrieved. The use of OPC communications standards on management domain tends to be comparable to SNMP approach. In [Westerinen and Bumpus, 2003] a longer version of this part of device management history is presented. Also in this work, the authors envision a future of distributed management based on web services protocols and common semantics and models.

When applying Web Services technology to devices and systems management, two concurrent specifications arose trying to cover in practice exactly the same domain: Web Services Distributed Management (WSDM), from HP, IBM, Grid community and other partners, and WS-Management, from Microsoft, Sun Microsystems, Advanced Micro Devices, Dell, and Intel. WSDM [OASIS, 2006b] is an OASIS standard since March 2005 foreseen to manage and monitor the status of web services. WSDM-compliant services can be monitored by a network management application that takes in account its status and performance indicators to determine which actions to carry out over currently

faulty or low performance services. WSDM is composed by two specifications: Management Using Web Services (MUWS) and Management Of Web Services (MOWS). The first one depicts how to define and access management interfaces of resources as web services, while the second one defines how to manage services as resources and how to describe the access those using MUWS. Web Services for Management, frequently simply known as WS-Management, is a SOAP-based protocol specification ratified by the Distributed Management Task Force organization as a Final Standard [Arora et al., 2004]. WS-Management offers a standard way for systems to access and exchange management data across the infrastructure network. The WS-Management specification allows more than get and retrieval of simple variables. The integrator is free to define its resource model as it best suits him and he can easily update and redeploy it on even during runtime. Due to openness regarding resource model, existing standards can be reused here, such as CIM-XML or OPC UA Information models for instance. This way, existing components models can be integrated in a complete SOA approach and accessible through an open web standard. As Netconf, WS-Management protocol was developed to substitute SNMP. Still, WS-Management, as WSDM, is prone to several remarks from the device management community concerning its drawbacks.

- "Overkill" strategy: too complex and resource-consuming when dealing with very simple applications.
- Critical subject: device configuration, monitoring and diagnosis are sometimes considered critical subjects to be dealt using web services technology.
- Open Resource Model: If every proprietary use its own resource model to each device, the interoperability becomes compromised – same issue as in SNMP approach, for example.

As the amount of information that needs to be exchanged increases, so does the efficiency of web services in comparison to SNMP as argued by [Pavlou et al., 2004]. Still, smaller amounts of data though results in higher traffic for web services. In this case, the performance of web services, in terms of coding and latency, is poor in comparison to SNMP for example. However, modern companies are demanding for agile and open solutions that should allow more sophisticated management procedures, which cannot be supported by traditional technologies, such as SNMP. Web services standards are now the brick technology across almost every enterprise ICT levels. By pushing this technology even to small low-resources devices level, it will promote a more integrated enterprise cross-level interoperability. The computational power is continuously increasing in ever smaller devices, what tends to eliminate performance issues in the future to come. Several implementations are already available in open source, such as Openwsman [Intel, 2009], Wiseman [Wiseman Group, 2008], and "WS-Management for DPWS" [SOA4D, 2009b]. In March 2006, the groups responsible for these two last specifications agreed to unify them into a single one, addressing this way important customers concerns as

referred by [IBM, 2006, Cohen et al., 2007]. However, no visible output came out of the harmonization efforts and the two specifications remained.

Open Management Infrastructure (OMI) [The Open Group, 2012], a recent open source project was created by The Open Group and Microsoft to provide a quality implementation of CIM and WBEM standards with a small footprint, high portability and performance possible to adapt to a wide range of management applications. These sets of implementations also include support for WS-Management, which validates this approach as an "all-terrain" solution for contemporary management service-oriented applications.

As discussed by [Pras and Martin-Flatin, 2007], one of the biggest difficulties in contemporary management solutions is that companies and end-users are prisoners of practices of an era when networks were small and easy to manage, resources were scarce, dependencies were not vastly dynamic and it was not that interesting for companies to integrate their complete ICT environment in an holistic form. The pervasive exploration of SOA turn out to be a good opportunity for deploying a clean and inventive approach. The adoption of web services for management purposes follow the software industry trend that is pushing this technology and creating a wide range of tools. The same author supports the idea that by exploiting SOA principles it will be possible to decompose management platforms into independent but still interoperable macro entities packaged as services. This approach can open new business models by customizing a management platform according to the end-user requirements and increase the competition among vendors and drive costs down. As in other related work, in [Papazoglou and Van den Heuvel, 2005] the authors survey some approaches used for web services management but it is once again clear that the scope is still business and enterprise level services and do not really adapt to industrial automation device level.

The current state-of-the-art in device management shows that a considerable effort was continuously placed on the advancement of management solutions in a more technical flank, being nevertheless still committed to former conventions inherited from previous solutions and entrenched due to the still current ubiquity of legacy equipment. Besides these, current solutions supported by SOA-based technology are still mostly focused on high-level enterprise and business environments and those that try to address the device level are only mostly dedicated to telecommunications or network control equipment. It is then evident that **an important research gap exists on how to push and adapt some of these SOA concepts, lessons and technology to enable a more effortless and standard management approach at device level in service-oriented industrial automation**.

3.2.3 Enterprise Integration

As shown in the study by [IDC, 2008], information integration is considered the biggest and most expensive task in the market for data integration and access software being expected to achieve 40% of the overall billing of $3.8 billion in 2012 for an average annual

growth rate of 8.7%. Enterprises are already exploiting some of the potential of SOA for some time now. In fact, this application domain had a big part of the responsibility on SOA, and particularly web services solutions, becoming increasingly popular. The ongoing next step is to expand this approach to all levels of the enterprise envisioning a direct and transparent interoperability along its different ICT layers avoiding unnecessary gateways and translation processes.

Many authors endorsed the application of SOA-based approaches to enable more advanced deployments on enterprise-wide integration. As depicted by [Vinoski, 2003] web services are proving in practice to be the key solution to deliver a transparent interaction between multiple enterprise middleware systems. An approach also supported by [Gorton and Liu, 2004]. As discussed by [Patrick, 2005] the impact of SOA on contemporary Enterprise Information Architectures (EIA) is transforming every data source into a service. The abstraction of retrieval processes, access supervision and security, along with the presentation of information in XML-based formats provides application developers a consistent approach to interface information sources. Nevertheless, the same authors refer that the developer still needs to create unified information views and execute data transformations. **Semantic web solutions such as ontologies and logic-based reasoning engines also promise to deliver an important contribution during these enterprise-wide integration tasks**. Current solutions are mainly focused on the use of standards to handle the integration problem. This approach recurrently fails as it does not scale into large range of applications and do not deliver the required level of agility to handle change. As discussed by [Panetto and Molina, 2008] semantic integration is an increasingly adopted approach to deal with heterogeneity within large and dynamic enterprise ICT systems. Integration is generally considered to go beyond mere interoperability to involve some degree of functional dependence.

Enterprise integration applications need to dynamically find, model and query data sources as enunciated by [Halevy et al., 2005]. Removing intermediary adapters if standard interfaces as web services are employed can facilitate these tasks. As also supported by [Bernstein and Haas, 2008], by avoiding broker and deploying the same protocol, such as web services throughout all system resources is the basis for the creation of a ESB solution. In [Vernadat, 2007], the authors promote SOA as a mean to sustain business process orchestration on top of the ESB. When building open systems architectures, it is fundamental to agree upon *de facto* standards as web services. Nevertheless, and although related SOA-based technology platforms may provide suitable solutions for technical interoperability, the "*absence of a scientific method to justify an enterprise architecture proposal and difficulty to evaluate and compare different architectures* " as argued by [Chen et al., 2008].

Today, intelligent automation systems based on distributed embedded devices integrate system intelligence in a limited amount of mostly monolithic computing resources accompanied by large numbers of resource-constrained devices. As already noticed by [Estrem, 2003], the incorporation of web services into EIA is a viable solution to support

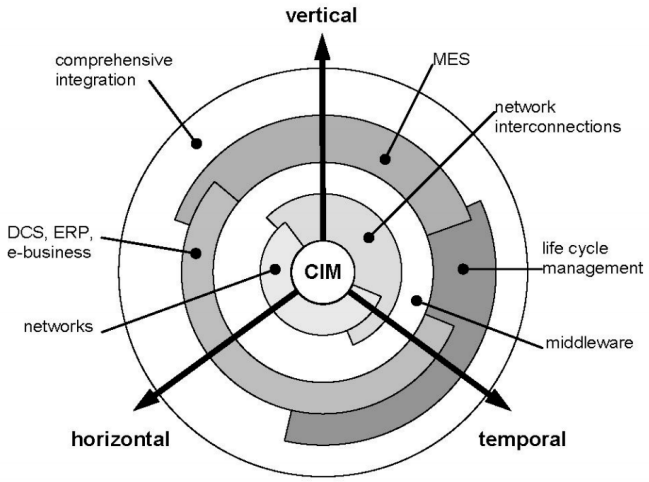


Figure 3.4: Model for communication and information technology integration in automation [Sauter, 2005]

the business needs and requirements of modern industrial automation domain. As illustrated by [Sauter, 2005], since the advent of Computer Integrated Manufacturing (CIM) in mid 1970s [Harrington, 1974, Ranky, 1986, Bedworth et al., 1991] the main goal is the integration of several disparate computer-assisted manufacturing subsystems into an all-inclusive and transparent enterprise-wide system. The original pyramidal model highly raised the enterprise expectations that ultimately were not met due to a big number of factors such as its complexity, technology availability or costs. The various facades of enterprise integration are not independent from one another, and most of the time they have not been even developed detachedly. As shown in Fig. 3.4 taken from [Sauter, 2005], they partially overlap, and they evolved in a bottom-up fashion.

The introduction of SOA approaches enabled the integration of these different aspects under the cover of a generic service interfaces that hides all implementation details – teams of service developers only need to agree on the interface details and functionality of their modules in order achieve mutual interoperability. In the domain of industrial automation, the work by [Kalogeras et al., 2006] introduced a SOA infrastructure allied to a set of tools to allow a vertical integration within industrial enterprises including the lower layers with soft requirement in terms of real-time. Although some advantages were observed such as absence of connectivity issues even in the presence of firewalls, the authors also enunciated some drawbacks in terms of trust and security what leads to an exponential use of specifications to cover these aspects. Following this same vision, the recognized work by [Karnouskos et al., 2007] proposed an web service-based integration with shop floor activities, in particular by promoting direct access to SOA-ready networked automation devices. When accessing a device from a higher level enterprise ICT layer some requirements should be taken into account: eventing support,

business process modelling and monitoring, *Heartbeat*-like feature, access to device status and metadata, and all these supported by a standard communication and information exchange mechanism such as web services. An effortless connection between factory floor and the high level decision services is essential to support more sophisticated and reliable decision-making. As also argued by [Karnouskos et al., 2007], the aggregation of fine-grained device services can be executed via orchestration in order to create increasingly coarser-grained services that offer a more refined and complex functionality to ERP systems. Most commercially available solutions assume intermediary elements, such as a MES or other translation gateways between the shop floor and the ERP system. This approach was further exploited during SOCRADES project [De Souza et al., 2008] and extended in [Spiess et al., 2009]. This methodology is even compatible with a cloud-based solution where each component could be hosted in different locations, inclusively in the cloud offering high-performance services, while other parts can be deployed locally on embedded systems at the device layer.

When connecting devices to the enterprise-wide ICT infrastructure there can be two distinct approaches (or both simultaneously): direct individual access to each device or access to a summarized information report provided by a data aggregator responsible for the monitoring of a group of devices, a production workstation or even a whole production plant. By connecting directly to the device, the application might need to deal with lower level details, pushing results interpretation logic to the ICT upper levels and boosting network traffic and possibly creating a "bottleneck" if a single node is responsible for collecting data from a large group of devices. Still, this approach allows a more detailed observation over the current status of the automation system . By having data aggregators (e.g one for each production workstation), these entities can retrieve local information from the devices, extrapolate relevant information and provide only summarized reports to upper levels. Although keeping data interpretation logic at shop-floor level, this way higher ICT levels will possibly not have access to detailed information about a particular device that it might be desirable in an atypical situation. An example would consist of a device that after a fault is detected, it can provide further details with different levels of granularity in accordance with the current fault type, like a debug mode. For a complete and robustness implementation, a **combination of both approaches would be more valuable**. During a troubleshooting phase, the system integrator can be assisted by the data concentrator summary report, requiring more specific details, if needed, by connecting directly to the faulty devices again employing a service-oriented approach.

Taking into account the current state-of-the-art on enterprise integration aspects applied to industrial automation domain, and particularly related with the integration of SOA-capable devices there are still some important aspects that should be further investigated. First of all, **there is need to determine the appropriate balance between the number and granularity of services embedded in low-resources devices versus the level of information encapsulation and aggregation methods by intermediate data collectors**. Associated with this latter topic, there is also a need to research and develop

new tools and methodologies on **how to process this big amount of collected information from device level and what to share with upper ICT layers in a SOA-compliant manner**.

This choice has a direct impact on how maintenance and diagnosis tasks will be conducted. Some work has already being conducted to enhance these procedures employing SOA-based approaches such as [Barata et al., 2007b,Zhao et al., 2010,Seilonen et al., 2011].

3.2.4 Semantic Reasoning

Firstly referred by Tim Berners-Lee in in [Berners-Lee and Fischetti, 1999], the Semantic Web is an on going Internet advancement in which the meaning of online information and services is clearly described and linked in an interconnected network of knowledge. The ultimate goal is to allow both people and machines to understand the information available in the Interned these days as discussed along several works such as [Hendler, 2001,Shadbolt et al., 2006,Daconta et al., 2009,Fensel et al., 2011a]. Supported by previous premises, the concept of Semantic Web Services was born as complement to traditional web services technologies as presented by [McIlraith et al., 2001,Fensel and Bussler, 2002, Payne and Lassila, 2004,Cardoso, 2007,Fensel et al., 2011b].

Since both SOA and Semantic Web concepts were originally conceived for high level business ICT, industrial automation applications are still rare and mostly prototypes. In fact, the application of these concepts can be considered transversal to many other domains such as those depicted in these sections related to SOA input.

Although approaching industrial automation domain, some approaches only tackle high-level communications, such as business-to-business relations. In [Terziyan and Zharko, 2003] the author addresses the semantic annotation of web resources and services to make them "understandable" by machines and aid during the discovery process in a peer-to-peer network environment. The authors of [Kulvatunyou et al., 2005] present a prototype employing semantic web technology to enhance the interoperability between manufacturing companies and contribute to an easier and low cost realization of virtual enterprises. The work by [Thramboulidis et al., 2008] presents a SOA framework supposed for embedded systems, although still relying on resource-intensive solutions such as Servlets which can constrain the deployment on low-power devices and a centralized UDDI element. An Integrated Development Environment (IDE) is also introduced to support the development and deployment based on IEC61499 [Vyatkin, 2007]. This IDE includes a dynamic Graphical User Interface (GUI) to support the management and update of the system ontology using SPARQL queries in order to ease up the integration process of new function blocks. In [Terziyan and Katasonov, 2009] the authors introduce the Global Understanding Environment (GUN) envisioning future applications of semantic web and agents into industrial automation. This middleware framework tries to provide a set of tools for building complex industrial systems consisting of components of different nature. Nevertheless the approach remains too abstract to be concretely

adapted to the current domain.

Other authors focused on the development of ontologies to describe the resources related with the industrial automation domain. Although not addressing SOA notions, the work by [Dibowski and Kabitzsch, 2008] delineated device descriptions based on OWL for automation devices including understandable semantics to support semantic retrieval of devices, semi-automatic parameterization and design of automation systems. The work by [Thamboulidis et al., 2007] promotes a device ontology inspired by [FIPA, 2001] specification and services as encapsulation of IEC61499 Functions Blocks (FB). This approach is still mostly focused on PC-range devices and relies on hierarchical brokers solutions such as UDDI. Other authors as [Schlenoff et al., 1998, Mönch and Stehli, 2003, Lin et al., 2004, Lohse et al., 2005, Lemaignan et al., 2006, Al-Safi and Vyatkin, 2007, Ribeiro et al., 2008d] delivered some ontology proposals to try to model the domain mostly in terms of types of equipment or process description that can be reused and merged when required.

In terms of ontology merging and mapping several solution are already available as developed by [Noy et al., 2000, McGuinness et al., 2000, Noy and Musen, 2003, Choi et al., 2006, Flahive et al., 2011]. In the roadmap by [Lastra and Delamer, 2006] the authors expose the insights and expectations from employing semantic web approaches to industrial automation domain. The authors defend the use of ontologies and semantic web services as mean to overcome known interoperability difficulties. The ultimate envisioned goal is to **enable automatic discovery, classification, composition and orchestration of manufacturing equipment**. This can be achieved by clearly stating resources semantics so that inference engines can use them to extract valuable information to facilitate previous goals. This approach is even more remarkable to ease the integration of new resources and processes and assist or even automate the interaction between the system resources. Other solution often referred is to simply rely on standards. However, as discussed by the same authors, the use of standards can introduce other limitations: abundance of standards and specifications that inversely constrain interoperability and integration, allied to the impossibility of standards to assimilate new entities that did not exist at the time of the standard development.

It is highly unlike that a universal solution for industrial automation will be ever available due to the multiplicity of domains plus the industry players desire to push their solutions encompassing a proprietary added value. On the contrary, the semantic web service can be incrementally implemented into current industrial automation installation to enhance autonomy, interoperability and agile reengineering.

In summary, by applying this approach to the industrial automation device level, components previously unknown can be identified and be ready to interoperate as a whole. It could be also possible to select the best available service following search parameters, such as QoS, service features, manufacturer, location, past activities, etc. After identifying the services, it is possible to compose them into complex processes through

orchestration and choreography as depicted by [Laukkanen and Helin, 2003]. A posterior work by [Delamer, 2006] employed an upper ontology for semantic web services based on OWL-S merged with other domain ontologies in order to semantically model processes, the environment and the relation between them. This approach allowed the interpretation of semantic descriptions and support the update of the model along system lifecycle evolution. In [Delamer and Lastra, 2007] this approach was applied to create a composite manipulator ontology prototype. The work pursued by these authors comprised a semantic-capable infrastructure based on loosely coupled interactions and dynamic discovery. Although these kind of semantic proposals imply an additional initial effort for developing the necessary ontologies, the posterior effort required to integrate new devices and services is expected to be significantly inferior.

As referred by [Terziyan and Kononenko, 2003], without mature standards, proof and actually working industrial cases semantic web approaches have small chances to be adopted by industry. As pointed out by [Sauter, 2007]: "*the gap to bridge today not only concerns the linking of two different network types but also the interconnection of two different mindsets*". Nowadays, when developing technology integration projects, the biggest problem along system lifecycle regarding data handling in not the retrieval of the data itself but mostly importantly to understand the true meaning of it. The same authors foresee the increasing adoption of web services solutions to implement middleware packages. However, a service interface by itself is not sufficient to achieve complete interoperability – that is the reason why so many standardization activities exist in the industrial automation area.

As pictured above, there are some implementations exploiting some of the advantages of semantic web techniques and most particularly of semantic web services, though are still some issues that need further research. Most of the architectures are still centralized or highly rely on brokers, do not attain industrial constrains in terms of real-time, reliability, or memory footprint, besides the expected mismatch of concepts and relations due to the assortment of concurrent ontologies from different suppliers and general suspicion from industrial automation staff. Consequently it is still recommended to take a stepwise approach by specifying solutions enriched with these new concepts for a small set of well known issues, verify its feasibility and breakthrough points to then clearly exploit the results.

In the scope of industrial automation device level, **the semantic approach should mostly focus on solving integration problems and assist or even automate the interaction with newly plugged devices**. To achieve it there is a need to define first an ontology capable to characterize a service-oriented automation system and the relations between its different elements, particularly devices and services. The ontology will provide structure and organization to the domain knowledge and describe both common concepts and relations between them. It can ease the analysis of features and configuration parameters inherent to a type of device, and also allow the reuse of models, definitions and relations into potential new applications.

During hardware analysis, a semantic reasoner can assist the integrator during the discovery of the best available device that fits system requirements and constraints. By retrieving their semantic tags included in devices description it will be possible to identify it and retrieve its main features and supported services. For example, which kind of services a particular range of devices expose as built-in generic services, or which services can be aggregated to support a more complex task. This ontology is expected to grow and be refined alongside system evolution, since new services and relations can be constantly added and existing ones updated. The ontology will act as an updated reference knowledge model upon which more intelligent behaviour can be supported. In a services scale, a similar approach can be foreseen. Abstracted from implementation intricacies, the systems integrator can search for services available on the network assisted again by a semantic reasoner during the creation of the industrial application and compose the existing services in a perceptive graphical style.

Even if for the moment it is a critical subject to address with systems integration experts, it is possible to picture the exploitation of semantic tags during system run-time. During a device failure or maintenance action, the application would be capable of discovering and consume an "equivalent" service available on the network. While discovery implies service description and semantic tags comparison, the invocation might need some translation mechanism to adapt to a possible differing interface – services coming from different vendors or developers might differ in the interface although implement the exact same functionality. Here, **the research community still has several issues to address to really convince industrial automation companies to invest in this approach**, such as security, run-time constraints, robustness, etc. Most of the existing research work lack some clarification on the scope and concrete objective is usually implicit and do not clearly expose neither the problem chosen to be solved, neither the concrete advantages when comparing with previous approaches. As a first step, **the semantic input must be kept in the scope of systems integrator assistance**. Then, regarding run-time constraints and performance results, it can be envisaged to wisely automate certain common tasks to avoid the constant need of human intervention.

3.2.5 Simulation

Due to the dynamic and distributed nature of SOA the process of simulating a service-oriented application does not respect most traditional testing, debug, simulation and validation processes. As depicted by [Chen, 2007], SOA systems are distributed systems and their modelling and simulation must fall into the research and practice of distributed system modelling and simulation. This domain is considered defiant due to its intrinsical parallelism, non-determinism, communication, synchronization, and interlocking problems.

In terms of testing and as discussed by [Bertolino et al., 2011], a combination of offline and online approaches can be followed:

- Off-line testing: the common development process where each service developer implements and test its code against some facsimile services that would interact with his service.

- Admission testing: check if the services developed are complaint with deployment target requirements, specifications and software/hardware platforms.

- On-line testing and monitoring: create and launch test cases in the real execution infrastructure and monitor how services behave individually and as a whole.

Sometimes during the design of a SOA application it can become too complex to understand how a set of services are behaving altogether to achieve the expected result. For the case when the system is behaving errantly, this can even more complicated to detect the reason for that unexpected behaviour. Several research work has been conducted to assist these processes in the scope of SOA applications. In [DeFee, 2004], the authors introduce the SIMPROCESS analysis tool to improve business processes and it includes both modelling and simulation capabilities. However this approach requires the design of a XML model for every application and does not integrate with an application already available or in process of development. The main focus is not much on testing and simulation the SOA system, but mostly to correlate performance metrics regarding business process model. Having in mind that simulation can reduce the error margin and consequently reduce the cost of service-oriented deployments, in [Chen et al., 2006] the authors present the SOAr-DSGrid, a service-oriented architecture for executing distributed simulation on the Grid-based applications. This prototype relies on a component-based framework for ease of component and simulation development. Following a similar approach, the work by [Tsai et al., 2006b] addresses the simulation, development, and evaluation of large scale distributed systems. In [Sarjoughian et al., 2008] the authors present a Discrete Event System Specification (DEVS) simulation framework compliant with SOA to help the model and simulation of service-oriented applications. As referred by [Wieczorek et al., 2008] to enable automated verification and validation of SOA-based ERP systems it is essential to employ a formal specification of the interaction protocols between the composed services to provide an accurate and unambiguous description of the MEP.

Although previous work tackled some aspects related with SOA simulation in relatively generic forms, the large majority are simply prototypes and were implicitly created to a particular application domain and cannot be easily extended to the industrial automation device level. In this domain, simulation is an area where SOA principles can be employed to validate a priori industrial automation deployments or simply test and check some sections of the system for requirements compliancy. Some new efforts are starting to address certain areas of service-oriented industrial automation.

The work by [Cachapa et al., 2007] joined DPWS and Delmia Automation tool to deliver a service-oriented solution that simulated industrial automation system using 2D/3D models of production automation devices interfaced by services. The authors

applied this approach to three demonstrators: a single virtual device, a hybrid environment with one virtual and one real device, and to a set of multiple virtual devices and instantiation on reusable components. This relevant solution allowed the integration of 2D/3D engineering tools to enable service-oriented software/hardware in the loop. An industrial automation installation can be enacted virtually, simulated and validated without risking real hardware equipment. It would be possible to simulate a new part in parallel with the real system and check for its feasibility and behaviour prior to integrate it into the real system. Since each simulated and real device is interfaced using services, the effort of replacing one virtual device by its physical counterpart can be significantly reduced. A continuation of this work is presented in [Cachapa et al., 2011]. In [Milagaia, 2009], the author applied this concept to a conveyor system and extend the control aspects using a MAS system. A similar approach is depicted by [Kirkham et al., 2008]. Here the authors employ a Virtual Reality Modelling Language (VRML) tool to design a machine and its individual devices exploiting again the potential of DPWS. In [Karnouskos and Tariq, 2008] a simulation involving 5000 sensors interfaced by DPWS and monitored by a MAS is depicted. Although relevant to simulate the fitness of DPWS in environments with a huge number of devices, it consists of simply a prototype and not a simulation tool. In [Spiess et al., 2008] it was used a lightweight BPEL version to simulate processes through the orchestration of services embedded in devices. The work by [Yeung, 2007] discussed the application of a Communicating Sequential Processes (CSP)-based approach to verify services in business process design as BPEL and WS-CDL. The authors formalized WS-CDL and BPEL into CSP to be able to check for behavioral consistency based on the notion of traces-refinement in CSP. Other authors as [Mendes et al., 2009b] exploited the mathematical formalism of Petri-nets to simulate and validate the coordination of the services provided by distributed entities, although this approach can be difficult to scale into bigger and complex system if the logic is not distributed and encapsulated across several entities. In [Nylund and Andersson, 2010] a manufacturing installation is decomposed into micro, meso, and macro levels to apply a modelling and simulation approach. Also here the authors exploit 3D modelling to abstract the rules, motion and behaviour of virtual entities, although not much attention is put on SOA aspects. In [Lobov et al., 2008] a compination of DPWS and Timed Net Condition/Event Systems (TNCES) is presented to model and analyse services and their compositions.

As presented by [Byrne et al., 2010], the area of web-based simulation is growing, however factors such as network latency, GUI limitations, security vulnerability, application stability and licensing restrictions should also be taken into account. Assuming that future industrial automation installations are supported by services that interface process tasks, it would possible to create simulation nodes that present a compliant interface and running those even at the same time as the real system to check its validity. Therefore, **SOA can be used to develop simulation systems that support hardware in the loop**, which is a major advantage, since it becomes possible to use simultaneously

virtual and real devices during the simulation – clear separation between services interfaces and hardware. This approach can even leave the door open for the introduction of further augmented reality approaches. At device level in the industrial automation domain the most important aspect is **to provide means to simulate device interactions to check for compliancy and verify possible erroneous behaviour before deploying the system into the real equipment**. The introduction of 3D modelling tools can also provide an essential contribution to help the visual verification of the system. Although some work has already been conducted, the area of simulation for this particular domain still needs further investigation and development to give system integration the tools to proper simulate and analyse the service-oriented application being made.

3.2.6 Middleware

The definition of middleware is sometimes vague, confusing or even conflictuous when defined by experts from distinct ICT domains of application. Taking into account several available definitions, middleware can be defined as *a class of software technologies designed to help manage the complexity and heterogeneity inherent in distributed systems* [Bakken, 2001], or as the *computer software that supports communication and management of data in distributed applications* [Krakowiak, 2003] or even as *a software layer between the operating system and the applications that provides a higher degree of abstraction in distributed programming* [Bruneo et al., 2007]. The common topics that can be extracted from most of the middleware definitions relates to its "invisible" nature when supporting a standard way of doing things so that the user do not need to take care of minutiae that do not relate with the most important aspects of the application to develop. The middleware is also commonly referred as the "glue" that ties together the different parts of a complex distributed system.

In the context of this work, middleware relates to the use of software-based models and technology exploiting SOA principles and guidelines running embedded at device level to enable a more complex behaviour and transparent interaction between the several infrastructure elements.

This software layer lies between the operating system and the applications on every element that interacts in the same network. It provides services and methods to software applications beyond those available from the operating system. Middleware-related solutions provides the foundation upon which software developers can implement their applications by focusing on the actual purpose and avoiding dealing with lower level details.

A service-oriented middleware must offer the system integrator a set of methods and services that can ease the development and deployment of the application in the form of a set of interconnected services in an effortless and transparent manner as possible. Due to the specificity of industrial automation device level, all communication implementations should be encapsulated and presented in neat form as well as other related features such as discovery, identification and invocation of services or asynchronous MEP handling.

When addressing SOA-based technology in the context of industrial automation device level the major focus is pointed out to software that can be embedded and run on small industrial devices. Although other kinds of technology related with chip processors, memory, I/Os and communication interfaces are evidently required to build a device, the current section will only address software-related work. As discussed in the roadmap work by [Cannata et al., 2008], SOA is expected to deliver an important contribution to the embedded devices domain, particularly in terms of middleware capabilities.

In [Barisic et al., 2007] the authors survey the potential of SOA to become a key factor in embedded software development, but only refer Jini, OSGi, e UPnP technologies. Nevertheless, the authors consider that SOA promises to overcome some of the known system design issues in embedded development processes. In [Ricci et al., 2007] the embryonic simpA-WS framework is depicted based on Service Agents & Artifacts based Architecture (SA&A). Other solutions, as in [Lin and Panahi, 2010] although promoting real-time services, the hardware requirements (dual core Linux servers in this case) are still too high to be employed by lightweight devices. In this work the RT-SOA ESB middleware monitors the performance and reserves resources in advance for each service in the process to ensure its real-time feasibility. The work by [Biffl et al., 2009] introduces the Automation Service Bus (ASB) with the goal of integrating heterogeneous components for automation systems engineering similar to the ESB in business ICT using a SOA approach. The authors of [Scholz et al., 2009] present and discuss an embedded SOA (eSOA) concept based on the definition of an embedded service (eService). The paper describes a middleware platform that supports the execution and development of embedded network applications by employing model based code generation and a pattern based service composition model.

As portrayed by [Greenwood et al., 2007] some efforts are being made to integrate web services solution into MAS development packages. These include JADE WSIG [Greenwood, 2005], WS2JADE [Nguyen and Kowalczyk, 2007], AgentWeb [Omair Shafiq et al., 2005] or OWL-P [Desai et al., 2006]. In the domain of sensor networks some middleware solutions embedded some SOA inspiration such as OASiS [Kushwaha et al., 2007], Atlas [King et al., 2006], USEME [Caete et al., 2008], TinySOA [Avilés-López and García-Macías, 2009] or SensorGrid [Tham and Buyya, 2005]. A recurrent characteristic of these systems is their hierarchical network structure, in which data is more and more aggregated towards the root. This solution may introduce critical bottlenecks and single points of failure for control oriented applications involving a big number of sensors and actuators.

As shown in previous sections concerning SOA contributions in each subdomain of application, DPWS is a widely employed solution to offer SOA-based features at device level as also confirmed due to the increasing number of deployments since the original SIRENA prototype [Jammes and Smit, 2005a]. Contributions such as [Jammes and Smit, 2005b, Mendes et al., 2007, Karnouskos et al., 2007, Zeeb et al., 2007a, Cachapa et al., 2007, Lobov et al., 2008, Ribeiro et al., 2008b, Colombo and Karnouskos, 2009, Cândido

et al., 2011] among many others support DPWS as the key solution for most of research initiatives and prototypes. In a scope more close to current industry trends, other ordinarily employed solution is OPC UA [OPC Foundation, 2008], a web sevices-based version of the OPC protocols widely deployed across industrial platforms. Contributions by [Leitner and Mahnke, 2006,Hadlich, 2006,Hannelius et al., 2008,Schleipen, 2008,Grossmann et al., 2008,Cândido et al., 2010,Seilonen et al., 2011,Son and Yi, 2012] help demonstrating the impact and influence of OPC UA in industrial automation.

Since DPWS and OPC UA are the two most recognised software-related solutions to apply SOA properties at device level, the present work will mostly focus on these. Although none of these embody a truly middleware solution, they provide definitions and guidelines that can implemented with an embedded communication stack.

3.2.6.1 Devices Profile for Web Services

A proposal for using web services protocols for device networking, entitled "Devices Profile for Web Services", firstly submitted in May 2004, is currently a standard by the OASIS Web Services Discovery and Web Services Devices Profile Technical Committee, since June 2009. DPWS is a stack of web-based protocols and profile for devices, which defines two fundamental elements: the device and its hosted services. It may be further noted as example that DPWS is already natively supported by Microsoft operation systems since Windows Vista.

Devices play an important part in the discovery and metadata exchange protocols, while its hosted services provide the functional behaviour of the device and depend on their host for discovery.

Besides hosted services possible to be developed by the end-user, DPWS also specifies a set of infrastructure services (see Fig. 3.5):

- *Discovery services (WS-Discovery):* used by a device connected to a network to advertise itself and to discover other devices [OASIS, 2009b]. WS-Discovery uses User Datagram Protocol (UDP) and a multicast address to broadcast and listen to the discovery messages.
- *Metadata exchange services (WS-MetadataExchange):* provide dynamic access to devices hosted services and to their metadata, such as WSDL or XML Schema definitions.
- *Event publish/subscribe services (WS-Eventing):* allow other devices to subscribe to asynchronous messages (events) produced by a given vendor-defined service.

DPWS is built on top of the SOAP 1.2 standard, and relies on additional WS specifications, such as WS-Addressing and WS-Policy, to further constrain the SOAP messaging model. At the highest level, the messages correspond to vendor-specific actions and events. Messages are delivered using HTTP, Transmission Control Protocol (TCP) and UDP transport protocols (see Fig. 3.6).

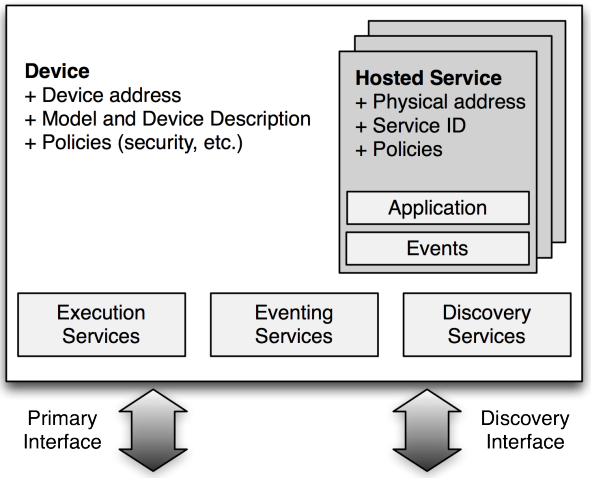


Figure 3.5: DPWS specification outline

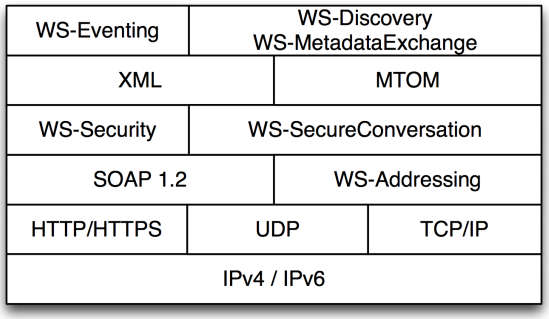


Figure 3.6: DPWS protocols stack

The most common behavioural pattern of a DPWS endpoint consists of discovering relevant devices in the network, retrieve the device description and its set of hosted services, invoke operations on selected services to interact with the device, and ultimately subscribe to available event sources. DPWS thus provides a small and efficient framework for peer-to-peer device interactions, fully compatible with the web services family of specifications.

3.2.6.2 OPC Unified Architecture

The OPC UA is the new version of the vastly deployed OPC architecture originally designed by the *OPC Foundation* to connect industrial devices to control and supervision applications as explained by [Hadlich, 2006, Leitner and Mahnke, 2006]. The focus of OPC is on getting access to large amounts of real-time data while ensuring performance constraints without disrupting the normal operation of the devices.

The original OPC specifications, based on Microsoft COM/DCOM, are becoming obsolete and are gradually being replaced by new interoperability standards, including web

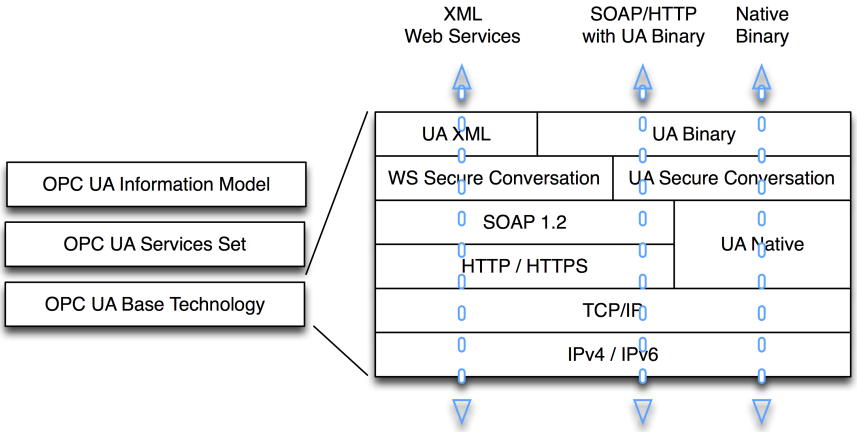


Figure 3.7: OPC Unified Architecture specification detail

services. This has led the OPC Foundation to publish a new architecture: OPC UA.

While the transition from original COM/DCOM to cross-platform capable web services is the most visible transformation, it is important to refer other significant ones. These comprise the support for secure communications, unification of several OPC data models, such as *Data Access*, *Alarms & Events* or *Historical Data Access*, as a single set of services, and extension to other domains such as manufacturing, production, maintenance and business applications.

As shown in Fig. 3.7, OPC UA can be mapped using widespread Web standards, including XML, WSDL, SOAP and other WS-* specifications, along with other particular OPC UA defined specifications.

In order to face original OPC security issues, current OPC UA specification already takes this lesson into account, although the introduction of security mechanisms may imply an impact on system performance. By allowing a flexible configuration it is possible to turn security parameters on and off according to application requirements. To support communication security, OPC UA specifies two alternative solutions: UA Native mapping and web services mapping. UA Native Secure Conversation is similar to TLS/SSL specifications comprising confidentiality, integrity, application authentication and secure channel functionality. When employing web services, WS-SecureConversation along with WS-Security, WS-Trust and XML Signature and Encryption specifications are used to support previous functionality. For readability purposes, these last specifications are implicitly hidden under WS-Secure Conversation element in Fig. 3.7.

In order to improve performance, additional specific protocols are used: OPC UA Binary encoding over UA Native, a TCP-based specific protocol. Transport protocols and information encodings can be combined in several ways, depending on the application requirements. Both UA XML and UA Binary messages can be transported by SOAP messages. In the latter case, the binary content is embedded in XML. UA Binary messages

can be also transported by UA Native messages.

OPC UA Information Model [OPC Foundation, 2006], as referred before, is a fundamental aspect since it unifies previously distinct OPC data models. The model uses a tree-based hierarchical representation, using references to relate different parts of the tree, thus providing a full-meshed network of nodes. Nodes in the tree can be of different types, including *objects* for structured representation and *variables* for dynamic data representation. This model can be accessed, browsed and manipulated through a set of generic services, which include services for establishing a (secure) communication channel and a session; services for reading, writing and subscribing to data; services for browsing, querying and modifying the tree representation; and services for calling arbitrary methods.

The OPC community does not dictate the detailed information model for every application. In order to achieve complete system interoperability, even using an interface specification with flexible query mechanisms, users in the same business space need to agree on some common information semantics. OPC UA strategy is focused on collaboration with major industry standards organizations and on how to move the information models without restrictions from these other industry standards organizations to an end-user community. These organizations include Electronic Device Description Language Cooperation Team (ECT), FDT Future Device Integration alliance (FDI), Machinery Information Management Open Systems Alliance (MIMOSA), GridWise, Building and Automation Controls Networks (BACnet), Instrumentation, Systems and Automation Society's Batch Control S88, ISA S95 and Open Modular Architecture Control (OMAC).

OPC UA thus provides a homogeneous and generic meta-model and defines a set of web services interfaces to represent and access both structure information and state information in a wide range of devices.

3.2.6.3 Other complementary WS–* specifications

Several sets of WS specifications are relevant to the previous two middleware specifications. It is then important to refer them according to their functional purposes.

3.2.6.3.1 Security

OPC UA relies on three OASIS standard specifications (WS-Security, WS-Trust and WS-SecureConversation) to ensure a secure channel between client and server. Although not formally part of the profile, those three specifications are natural extensions to DPWS in contexts where security is required.

3.2.6.3.2 Management

While OPC UA is particularly focused on resource management and supervision, DPWS defines a general-purpose, but extensible, architecture for service-oriented interoperability at the device level. One way to extend the capabilities of DPWS in the resource

management domain is to integrate it with a dedicated WS-based management specification, such as WS-Management [Arora et al., 2004]. As seen before, this specification is free to be extended beyond the original set of operations and it relies on a set of existing specifications, including WS-Addressing for endpoint descriptions, WS-Transfer and WS-Enumeration for resource description and access, WS-Eventing for event notifications and WS-Security for security.

3.2.6.3.3 Binary encoding

OPC UA defines a specific, non portable binary encoding for optimising message size and processing time when exchanging large data sets. While DPWS does not include any binary exchange format in the profile, the specification is open to allow this extension. World Wide Web Consortium (W3C) Efficient XML Interchange (EXI) standard coming for binary XML encoding emerges as a strong candidate to deliver concrete performance improvements, as depicted in [Moritz et al., 2010].

3.3 Services Granularity

As defined by [Schmelzer, 2006] "*granularity is a relative measure of how broad a required piece of functionality must be in order to address the need at hand*". Consequently, the service designer must find the right balance of fine-grained and coarse-grained services to meet the evolving system requirements. Most of the research done in the domain of granularity, such as the work conducted by [Herzum and Sims, 2000, Mili et al., 2001, Wang et al., 2005] is mostly focused on measuring and assessing the impact of granularity of components. It is also common to assume that component granularity is defined recursively based on the composition of finer-grained components. Other authors as [Sims, 2005] measure the level of granularity by counting the number of components invoked within a service, by counting the number of operations available in a component, or by counting the number of database tables updated.

In SOA, as well as in most of the distributed applications and others, the choice of granularity is one of the most subjective topics to address and it is always constrained by each user experience and domain of application. This issue is even more significant at design phase when the systems integrator is modelling the application for the first time and needs to choose the adequate granularity for that current use-case, e.g. **should a sensor be a service itself or simply the equipment that encompasses that sensor should be presented as a service?**

The overall quantity of functionality encapsulated by a service determines the service granularity. For instance, a service based on an entity service model will have a functional context associated with one or more related entities. Functionality associated with the chosen entity belongs within the service functional boundary. The larger the quantity of related functionality, the coarser the service granularity. Then, services with narrower

or targeted functional contexts will tend to have a thinner grained level of service granularity. Note that the level of service granularity is set by the service functional context, not by the actual amount of functionality that resides within the physically implemented service. The data granularity is determined by the quantity of data exchanged during service execution. For example, a service that retrieves a complex machine status will have a coarser level of data granularity that a service that simply retrieves the motor status present is that same equipment.

Most times it is very challenging and time consuming to define the fittest granularity for a service to adapt to a particular application. Fine-grained services address small units of functionality or exchange small amounts of data. If it is designed too small, there is a risk to develop a useless service (if considered alone) since it must be composed with others to have an application meaning – need to shift some business logic into upper layers. This leads to an application with uncountable very small services with the real application logic totally implemented using an orchestration- or choreography-based solution. Fine-grained web services are a direct mapping of more traditional practices in management: each web service operates at the managed object level. As discussed by [Pras and Martin-Flatin, 2007], this is the approach currently adopted by standards bodies such as in WSDM and WS-Management by default, if extension options are not considered. Although it is efficient to transfer many compressed managed objects in bulk, unfortunately it does not really comply with SOA vision by definition.

On the other endpoint, services too big can become less reusable. If a service cannot absorb requirement changes through configuration then its granularity must be diminished. Coarse-grained services can be considered "more" SOA compliant and work at a higher level of abstraction. However, by encapsulating a larger amount of functionality within a particular abstracted interface and thus reducing the number of service requests necessary to accomplish a task there is a risk of returning an unnecessary load of data and difficult futures services updates to meet new system requirements. Each service should relate to a high-level task or to a macro-component of the service-oriented management application. Coarse-grained services are by definition more autonomous encompassing their own kind of functionality and are loosely coupled regarding the rest of the system. As also stated by [Pras and Martin-Flatin, 2007], **the performance of coarse-grained services for constructing management solutions is still an open issue that deserves further research**. A similar problem occurred for CORBA and J2EE in the past and adequate solutions appeared a few years later. A summary of the advantages and drawbacks on employing course- or fine-grained services is shown in Table 3.3.

In [Haesen et al., 2008], the author besides discussing the impact of granularity on performance, reusability and flexibility, he also provides a relevant classification schema for service granularity (see Fig. 3.8). About data granularity, the author makes a distinction between data that is sent to the service (input data granularity) and data that is returned by the service (output data granularity). For functionality granularity, the previous author distinguishes between the amount of functionality that is always offered

Coarse-grained	
Advantages	*Drawbacks*
+ All data encapsulated in a single request	- Complex data, large message size and internal service errors more difficult to track
+ Message carries complete context	- Can lead to false sense of state and requires sending complete context for each request
+ Self-described and self-contained	- Can be too specific for a current scenario and not reusable in others
Fine-grained	
Advantages	*Drawbacks*
+ Small messages containing simple data	- Increased network traffic and overheads by dealing with multiple service requests
	- Absence of context
	- Might reflect closely current implementation
+ Individual services can be composed into higher level services	- The user must understand the logic behind each process (sequencing, pre-post conditions, error handling, etc.)

Table 3.3: Comparison of fine- and coarse-grained services (updated from [Wilkes and Veryard, 2004])

when calling the service (default functionality granularity) and the functionality that can optionally be configured (parameterised functionality granularity). Business value granularity indicates the extent of the added business value provided by a service.

To be completely effective to a systems integrator, service interfaces should clearly expose the operations they perform as well as the required input and output parameters, along possible errors or exception handling. Service interfaces should be easily understood by humans and at the same time possible to be used by data accumulators and semantic reasoning engines to extract all necessary information. The granularity should be in adequacy with the considered application and be permissive to support agile reconfiguration.

The choice of granularity is especially committed to the physical topology of the devices that compose the system and the functionality they are expected to offer to the

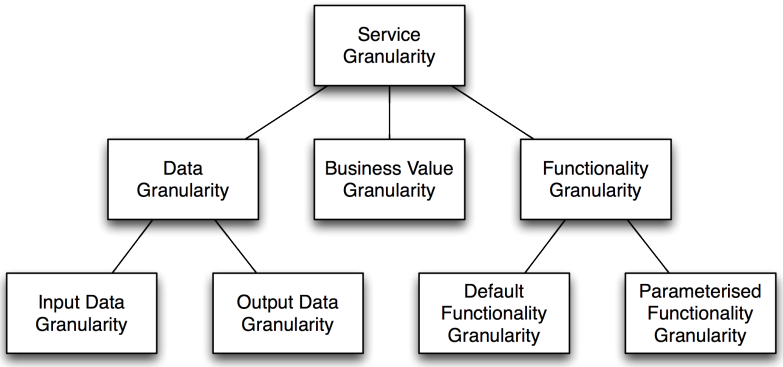


Figure 3.8: Types of service granularity according to [Haesen et al., 2008]

system as whole. As referred before, the granularity when employing a service-oriented approach into industrial automation device level should take into account low-level requirements in terms of real-time, reliability and safety. Whenever these requirement inhibit the use of SOA-based technology at this level, the granularity level should increase to the level it will be possible to encapsulate this resource below a service interface and at the same time warrant these requirements. **The service must then expose each resource in a more application- or process-oriented manner**, i.e. present the tasks it is able to perform in the current context.

In [Haesen et al., 2008], the authors also upholds that "*even if the importance of coarse-grained services is often stated, enterprise architects nowadays have to deal with a broad spectrum of possible service granularity levels for different granularity types*". After discussing the granularity problematic in general terms in the scope of SOA and automation domain, it is clear that **both coarse- and fine-grained services are necessary at device level in industrial automation to handle different system requirements**. Though course-grained services can deliver higher-level information or execute more complex tasks, fine-grained services are particularly essential for particular situations such as debug and maintenance activities.

3.4 Discussion

The present chapter investigates the contributions that SOA can provide to the industrial automation domain, in particular to its device level. Based on the existing related work and domain contemporary requirements these contributions were segmented into six sub-domains, respectively High Level Control, Device Management, Enterprise Integration, Semantic Reasoning, Simulation and Middleware. Even if some of the areas might overlap in some particular aspects, such as the introduction of semantic reasoning in enterprise integration or simulation for service-oriented control, these topics cover all the major investigation vectors currently active concerning the introduction of SOA principles and technology into industrial automation. The current state-of-the-art confirms that there are still domain branches that require further research to achieve consequent valuable contributions.

Concerning real-time web services solutions they are still rare, especially for industrial automation device level and are not widely deployed or even recognized. Nevertheless, it is acceptable to say that it will be soon possible to deliver real-time performances that compete against current conventional industrial automation solution, in particular fieldbus-based technology. This statement is even more valid due to the fact that hardware resources prices are decreasing for an increase of CPU and memory. Regarding the service-oriented control approaches (orchestration versus choreography) it is evident that neither one of these approaches will arise as the ultimate solution, although orchestration specifications such as BPEL are getting a lot of attention in this moment. If no real-time web services solution is available for a time- or safety-critical control setting,

service-based interactions should be simply kept within the scope of the device access and management. From a service-oriented application point of view, services should act as an endpoint to execute more abstract tasks (process-oriented) where the service interface clearly states its function and output instead of a set of unreadable I/O values.

Regarding enterprise-wide integration using SOA, some advantages are immediate such as the reduction of connectivity issues even in the presence of firewalls, although some authors refer some shortages in terms of trust and security. This situation is leading to an exponential invention and deployment of specifications to cover these aspects. Regarding the access to device level, two distinct approaches are currently put in place: direct access to the individual device itself or through information aggregators to collect and process data from devices in an incremental way along the system layers. As discussed before, for a complete and more adaptive solution, a combination of both approaches would be more valuable. As example, during a debug or maintenance intervention a system integrator can be assisted by a summarized activity report, requiring more specific details, if needed, by accessing directly to the actual devices. This way there is an urge to define the appropriate balance of services granularity embedded in low-resources devices versus the level of information encapsulation and aggregation methods by intermediate entities. Associated with this subject, the systems integrator requires new tools and methodologies on how to retrieve and process a big volume information from device level and what to share with upper ICT layers in a SOA-compliant manner.

A service interface by itself is not sufficient to achieve complete interoperability – that is the reason why so many standardization activities exist in the industrial automation area. However, the misuse of standardization can introduce other limitations: proliferation of standards and specifications that inversely constrain interoperability and integration. Furthermore, standards are sometimes not able to assimilate new concepts or situations that were not evident during the development of the actual standard. It is highly implausible that a universal solution for industrial automation will be ever available due to the multiplicity of domains plus the industry players desire to push their solutions encompassing proprietary added value. A complementary solution can be achieved through the employment of a semantic web inspired techniques. Semantic web services can be incrementally implemented into current industrial automation installations to enhance the autonomy, interoperability and agile reengineering tasks. At device level, the semantic approach should mostly focus on solving integration problems and assist or even automate the interaction with newly plugged devices and services. An ontology modelling the different aspects of a service-oriented industrial application will provide structure and organization to the domain knowledge and describe both common concepts and relations between devices and services. During a device failure or maintenance action, a semantic assistant would be capable of facilitating the discovery and consumption of an "equivalent" service hosted by a different device. While discovery implies service description and semantic tags comparison, the invocation will required a translation process to adapt to a possible divergent interface – services coming from

different vendors or developers might differ in the interface although implementing the exact same functionality.

Regarding simulation aspects, SOA can be used to develop simulation systems that seamlessly support hardware in the loop which is a major advantage since it becomes possible to integrate simultaneously virtual and real devices during simulation – clear separation between service interfaces and hardware platform. In the context of a service-oriented application in industrial automation the most important aspect is to provide the means to simulate device interactions, check them for requirements compliancy and verify possible erroneous behaviour before deploying the system into the real equipment.

The present state-of-the-art regarding SOA-based software for industrial automation device level confirms that DPWS and OPC UA specifications are the two most endorsed solutions. However, none of these specification on its current state can alone entirely fulfil the requirements of device-level SOA, and a combined approach can present vital benefits to the domain. SOA paradigm is increasingly ubiquous across the most disparate domains of application, then it is important to continue the research on how to map SOA principles and methodologies onto device level in the service-oriented industrial domain in order to achieve a cross-layer unification. Because these two specifications are currently the most employed when there is a need to push SOA into the industrial automation device level a more extensive analysis and discussion is presented in Chapter 5 – section 5.4.

Even if it is clearly a subjective topic when designing service-oriented applications, services granularity must be further discussed and guidelines must be specified to enable a more coherent application organization. In industrial automation, the choice of granularity is especially committed to the physical topology of the devices that compose the system and the role they are expected to assume on the industrial process. Although SOA principles favour course-grained services, in industrial automation both coarse- and fine-grained services are required at device level to better handle reengineering aspects and support customized adjustments along device lifecycle. Though course-grained services can deliver higher-level information or execute more complex tasks, fine-grained services are particularly essential for particular situations as during debug and maintenance activities.

A summary of these research opportunities is presented in Table 3.4. The current work does not cover all of these areas of interest and it is more focused on the areas of device management, semantic reasoning and middleware.

It is a reality that the SOA paradigm is increasingly ubiquous across the most disparate domains of application. In a broader sense, a strategic research challenge is on

Area of Interest	Research Opportunities
High Level Control	+ Real-time web services supporting some of the industrial-domain features already supported by latest field buses specifications. + Application to process-level control exploiting standard and abstract service interface.
Device Management	+ Service-oriented architecture for device lifecycle support. + Service-oriented device model to enhance agility at device level by improving device on-demand and on-time customisation. + Enhancements on device access and manageability through openness and standardization.
Enterprise Integration	+ Enterprise cross-layer interoperability. + Data mining and aggregation. + Intelligent monitoring and reporting. + Service granularity methodology and code-of-conduct.
Semantic Reasoning	+ Assistance during Reengineering / Maintenance / Diagnosis interventions. + Semantically-compliant industrial automation devices. + Domain ontology support by major industry players and academia.
Simulation	+ Hardware in-the-loop solutions. + Simulation of device interactions, including both real and virtual instances.
Middleware	+ Merge of SOA and industrial automation principles, requirements and expectations. + Merge of efforts, particularly regarding DPWS and OPC UA specifications.

Table 3.4: Summary of major research opportunities concerning SOA application to Industrial Automation

how to transfer the SOA principles, methodology and technology into the scope of systems integrators to achieve an enterprise cross-layer unification and assist agile reengineering interventions without completely obliterating the acquired know-how. A reference architecture is needed to assist systems integrator device level role along the lifecycle of the actual service-oriented application . Semantic web solutions such as ontologies and logic-based reasoning engines are considered agile factors to enhance enterprise integration tasks and can be extended to device level reengineering assistance and possible automation. As a first step, the semantic web input must be kept in the scope of systems integrator assistance. Then, regarding run-time constraints, accuracy and performance results, it can be envisaged to wisely automate certain common tasks to avoid the constant need of human intervention. In the reviewed work there is no device level model that entirely addresses industrial automation device level in a SOA context without compromising both domains principles and requirements. Most of the existing deployments rely on mostly hard-wired implementations done to test or support parallel validation goals. A combination of the requirements, constraints and values of both domains can contribute to a good compromise and enable a further exploitation of SOA in industrial automation device level.

At the same time as web services and related SOA-based technology platforms continue to deliver improved solutions for technical interoperability, some authors as [Chen et al., 2008] alert that there is still no scientific approach to evaluate an architecture proposal in this domain being this way arduous to compare and assess two concurrent solutions. Each author must then first verify its feasibility and fitness to the application

domain before performing the appropriate testing and comparison with previous solutions results and judgements from domain experts.

4

Device Lifecycle Support Architecture

4.1 Introduction

Having in mind the scope of application of SOA in an industrial automation device level context depicted along previous chapter, the present reference architecture includes a set of elements that can be combined to better adapt to particular system requirements and encompass the different aspects of device lifecycle support.

In a service-oriented system, all elements expose their abilities in the form of services. Besides the need to easily deploy services into devices, there is an emergent need to assist, and possibly automate, the process of identification, setup and management of these resources in an agile manner. These aspects are strongly related with the implemented system processes that are executed by services hosted in the according devices.

In general terms, the present architecture exposes elements and methodologies to increase devices interoperability and agility performance at device level in service-oriented industrial automation domain. Device interoperability and subsequent overall system agility can be enhanced through the employment of semantic techniques on top of a collection of devices operating in a service-oriented industrial domain able to discover, recognize and process available information to assist or automate certain integration or reengineering tasks. In this setting, the systems integrator is assisted by intelligent entities and tools to complete his assignments in a more agile manner.

Some previous work, such as [Van Brussel et al., 1998, Gunasekaran, 1998, Barata, 2004, Leitão and Restivo, 2006] already addressed shop floor agility, however the current

work explores a SOA approach at device level to solve mostly integration and application deployment issues during reengineering interventions. The present reference architecture work follows the premise that all devices are compliant with a SOA approach as further detailed in Chapter 5. Concerning legacy equipment, a software-wrapper solution is also proposed.

4.2 Architecture

The current architecture was created by combining entities that can mutually interact and be combined to better support system evolutive process in an agile form. The reason to define a distributed collection of components is to allow a customized mould according to system requirements, available device capacity, application context, cost and performance constraints. All architecture elements are abstracted as services in order to ease their own integration into a lifecycle support infrastructure as well as its interaction with service-oriented capable devices.

As a first stage, it is important to focus on the elements related to device management that will aid the systems integrator every time there is a need to connect to a device to perform a certain task. These interventions can be related with commissioning integration aspects or lifecycle reengineering assignments. As a second stage, it is relevant to focus on the system process aspects and how the different devices and services interact to achieve the expected objectives. During system lifespan there is a need to update the existing process plan due to a wide variety of aspects such as adjustment of requirements, equipment modifications or detected misbehaviour.

An outline of the architecture with its main components is represented in Fig. 4.1. The following subsections will present in more detail the different elements of this architecture.

4.2.1 Service

In this context, a service is a software component that encapsulates a function or behaviour under its interface following SOA principles of design and technology. It can be discovered and exploited by other network entity in order to execute a particular task or to retrieve some kind of information. The service is the key element responsible for bonding all the architecture elements and enable a transparent interaction between them.

4.2.2 Device

A device, or more specifically a logical device, represents the entity that abstracts an application element, while its hosted services embody the functionalities that it allows others to exploit in the actual context. These can be employed to abstract the physical device upon which they were deployed. Abstracted by a logical device, the physical device will already embed some built-in services that will allow the deployment of applications, but

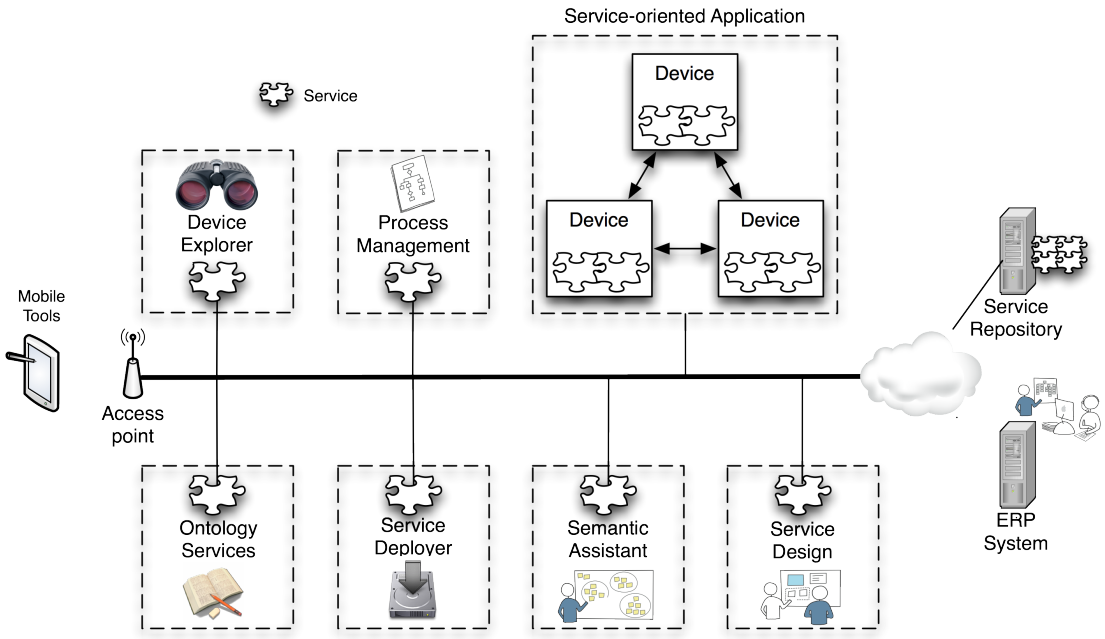


Figure 4.1: Service-oriented device lifecycle support infrastructure

also further added-value services to support identification, setup, monitoring and diagnosis of that particular device. This set of services is built-in the firmware by the device builder, being immediately available when taking a new device out from the box and plugging it into the network. This collection of services is considered generic to every particular range of devices. The ability to deploy a service is itself supported by a built-in service, which allows the systems integrator to deploy its own resources i.e. logical devices and its services. In a service-oriented production system, user applications are dynamically deployed in a form of logical devices that interact between them through their hosted services with no a priori imposed control architecture. An application can even be composed of several of these logical devices into several layers of increasing abstraction, in an orchestration or choreography manner – application construction based on existing building blocks. Chapter 5 – Service-oriented Device Model will present in more detail the proposed device model.

4.2.2.1 Legacy Equipment

As in any other ICT domain where a new approach is being introduced, there is a recurrent need to integrate legacy equipment that does not immediately comply with the new solution and at the same time it cannot be removed or replaced. In these situations, the common procedure is to implement a software wrapper solution that will allow this legacy device to exist and communicate in the new system. This method is executed by

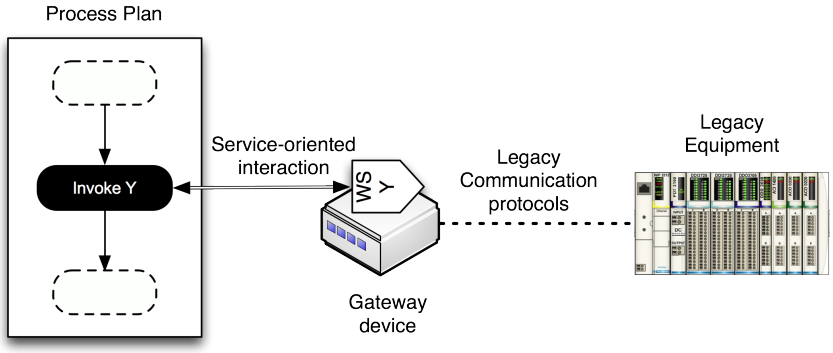


Figure 4.2: Software-wrapper solution for legacy equipment

encapsulating previous applications within a layer of the new technology and performs all the needed software translation and mappings between these.

In the scope of the proposed architecture, the first step of the integration process is to investigate the legacy device information, features and what tasks is it able to offer to the system. The second step is to emulate the existing device into a logical device deployed into a gateway device responsible for executing the required translation tasks and to communicate with the legacy device using the previous communication norms (see Fig. 4.2). This new device will include the applicable metadata information to allow it to be discovered and identified in the network. It will then be available as any other compliant device and present its own services to the system. As example, whenever a service is invoked the actual implementation will translate this call into the legacy protocols or technology and process its output, when needed, before sending it back to the invoker.

4.2.3 Service-oriented Application

A service-oriented application is the result of a composition of several resources that cooperate between them exploiting the services each one offer in a coordinated routine. The control configuration is not strictly predefined and can be constructed based on available building blocks – logical devices and services. In a holistic overview, the application will emerge from the composition of simpler modules to progressively create more complex structures and behaviours. The system integrator is responsible for modelling the interaction patterns between these in order to achieve a behaviour in accordance with current system requirements and constraints – a process plan.

4.2.4 Ontology Services

The provision of a clear and established domain representation of the device level in service-oriented industrial automation requires a specification of a model that will detail the concepts and relations of the existing entities. As defined by [Gruber, 1993], an

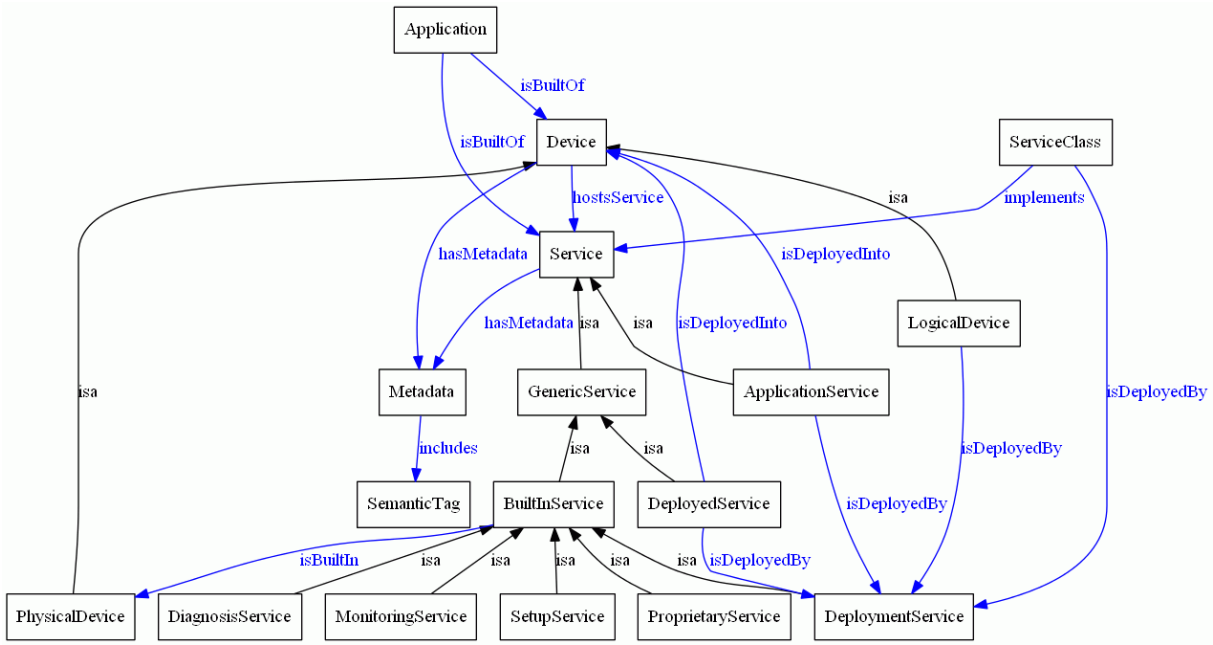


Figure 4.3: OWL Device level ontology overview

ontology is *a specification of a conceptualization. That is, an ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents.* In this context, this ontology is always evolving due to the fact that new services, metadata, specification and relation patterns are added or modified during system lifecycle. The ontology will act as an updated reference knowledge model upon which more complex processes can be supported.

In the scope of this work, an ontology was developed to specify the resources involved in a service-oriented application and their mutual relations. The ontology specifies the available resources in a service-oriented application and defines how these resources relate with each other to shape the current implementation. By retrieving an ontology individual it is possible to determine what are its relations in the associated context. As depicted in Fig. 4.3, the ontology simply focus on how the service-oriented entities are related and does not attempt to characterize the industrial automation specificity in terms of physical equipment or practices. The current ontology depicts the entities and relations at device level and models the different aspects of the proposed device model further detailed in chapter 5. In this last area of research, several proposals such as the ones from [Schlenoff et al., 1998, Mönch and Stehli, 2003, Lohse et al., 2005, Lemaignan et al., 2006, Lin and Harding, 2007, Ribeiro et al., 2008d] can be employed to extend the present one.

Due to the adjustable nature of an industrial automation application it is expected that the existing ontology will need to be updated along system lifecycle, particularly on what concerns current system individuals, i.e. ontology classes instances that refer to

an existing resource of the system. The proposed architecture takes this into account by exposing the management of the ontology as a set of services available in the network. Not simply services to set, update or retrieve ontology information, but it can be also used to deliver the collection of information used by the Semantic Assistant to provide a more complex and specific output.

4.2.5 Service Deployer

The Service Deployer represents a service-oriented software component that will support the deployment of services into physical devices available in the network. This component offers this function to the network in the form of a service possible to be employed whenever there is a need to update the set of resources deployed in a particular physical device or metadata modifications.

This resource is mostly used by other architecture elements and is supported by the existence of deployment built-in service in each device as detailed in the Chapter 5.

4.2.6 Service Design Tool

This architecture element refers to a service-oriented IDE chosen by systems integrator to design or reengineer the present set of devices and services within the current service-oriented application. This element will include not simply a traditional set of development tools to enable the implementation of services using a technical solution that fits current application context, but should also support the automated adaptation of these resources so that they can be deployed using the services offered by the Service Deployer.

4.2.7 Semantic Assistant

This element does not only support the execution of semantic reasoning tasks based on the current state of the system ontology, but also exposes these functionalities in the network as services. These services can include the semantic identification of devices and services, retrieval of generic services for a range of devices, mapping of features and services, service translation, etc.

This element can provide an important assistance to service design tools during the development of new logical devices and services by ensuring that it remains compliant with infrastructure knowledge model, system specifications and expected QoS. The Semantic Assistant can aid the integrator by advising which services should be deployed into that particular device taking into account the type of device, context and history records.

The Semantic Assistant can be configured to behave according to different levels of autonomy depending on the system critical factor set by the systems integrator. As a first step, the semantic input must be kept in the scope of systems integrator assistance. Then, regarding run-time performance and results, it can be envisaged to gradually automate certain common tasks to avoid constant need of human intervention.

4.2.8 Device Explorer

This element is a software component mostly used during analysis, setup and troubleshooting of a service-oriented application. It comprises a perceptive Human-Machine Interface (HMI) that allows the integrator to search the network for available devices and services, check their status, visualize metadata, or test available services. Some standalone explorers, such as the one presented by [Zeeb et al., 2007b] already supports the above feature plus search filters, discovery proxy and dynamic invocation of services.

Still, at device level in service-oriented automation systems other features are expected to deliver an important input along system lifecycle. Features such as the retrieval of current devices topology, deployment of semantic translation solutions, access to the Semantic Assistant to easily identify devices and improve posterior interactions, such as for configuring and managing devices, or deployment of resources into devices. Once again, by exploiting the Service Deployer component it will be possible to deploy saved resources from the Device Explorer itself into available physical devices. The Device Explorer simply needs to invoke the deployment service, passing the service resources to deploy and the references of the device where to deploy. Without changing of HMI, the integrator can then test if deployed services are running properly. Also, by default, the Device Explorer can subscribe to the *Heartbeating* event-based services available in each device present in the network in order to warn the systems integrator when a device goes offline. More details on use-cases involving this and other architecture elements will be further detailed in section 4.3.

4.2.9 Service Repository

This element can be seen as a repository of logical devices and services that can be stored and downloaded whenever they are needed. A Service Repository can be either locally based or available online as a business web portal. The online repository can be updated with resources submitted by third-party developers and made available online for the community that wants to employ them in their own systems. These resources may include functionalities particular to diverse domains, enhanced features for setup or access of information, particular equipment implementations or communication gateways. The integrator simply needs to search the catalogue for the ones that best fit the current application, download them to the local repository and deploy them into the appropriated physical devices. For example, when built-in services are not enough to handle particular system requirements, there is a need to deploy extra services that might allow a more complex control or management over the device.

This approach promises to open new business models since any developer can now create its own resources, publish them online and make it available to the community to use them either for free or by paying a fee to cover the cost of development. By promoting competition between service developers, the end-user will gain with the increasing

added value, robustness, performance and availability of these. Moreover, the clear separation between hardware and software layers will allow the same service to be deployed into different physical devices while being seamlessly available in the system. Even if implementation changes due to variations on device firmware, the service interface remains the same – a service can be substituted by an equivalent one without changing anything only by maintaining the service interface.

4.2.10 Access Point

The Access Point is a communication gateway for wireless devices to the remaining network infrastructure. Since the complete infrastructure is expected to relay on open standards for communication, both wireless and wired resources are able to interact transparently.

In the current context, wireless devices are mostly envisaged to be employed during commissioning, troubleshooting, maintenance and diagnosis processes since it will allow the mobility of the systems integrator along the shop floor to check locally the device status or system behaviour. Whenever there is a need to interact with a particular device for configuration, maintenance, monitoring or diagnosis purposes, a portable wireless device can be used, such as a smart phone or a tablet with wireless access, running the required Process Management tools or a Device Explorer. Shop floor devices are embedded with generic built-in services that will provide basic discovery, management and control operations. In the same way, the actual deployed application will be easily discoverable and directly interoperable in the network since it is deployed in the form of several logical devices distributed across different physical devices.

4.2.11 Process Management Tools

For every industrial automation installation there is a need to define the set of processes to be executed by the system during run-time. In the context of a service-oriented application, a process consists of a flow of invocations of services or events handlers provided by the available shop floor devices in a predefined way to achieve production goals. This element is responsible for creating and managing these processes supported by a graphical interface that promotes a visual combination of resources and their interaction patterns to create these processes.

To create a more stable SOA application, design models such as Service Component Architecture (SCA) as defined by [Beisiegel et al., 2005] emerge as a reference specification model. Assuming a continuous need for reengineer, more flexible and widely exploited service-oriented process management specifications as WS-BPEL [Alves et al., 2006] should be used. As for other architecture elements, these tools can benefit from the collaboration with other elements to assist systems integrator assessments and developments, as presented in sections 4.3.6.

4.2.12 ERP System

An ERP system is enterprise wide information system which consolidate information from various functions or departments of an organization. With the advances of SOA in the business level ICT, most of the solutions already available implement or support a service-oriented approach to interconnect its different divisions. By pushing the boundaries of SOA to industrial automation device level the connection between these two areas is expected to become more transparent and reduce the number of software gateways between them.

4.3 Use-case scenarios

As referred before, the proposed architecture is constituted by a distributed set of elements that can be composed to interact in several ways to fit current system requirements. The present section identifies the uses cases supported by the proposed architecture in the context of industrial automation device level to assist and agile system lifecycle interventions. In the following use cases scenarios the principal actor is the systems integrator and focus on what can be achieved by exploiting the architecture added value. These use cases were created based on the identified functional requirements for device lifecycle support in a service-oriented industrial automation installation and intend to demonstrate how a systems integrator will benefit from exploiting it.

4.3.1 Ontology Services

In distributed infrastructures it is of major importance to ensure information consistency across the environment. Methodologies and guidelines to support it must be applied when there are distributed resources that make use of or recurrently update the system ontology. The access to the system ontology is also interfaced as a service so that in the context of the device lifecycle support architecture it represents another building block that can be plugged whenever needed to provide precise and updated knowledge about the current system.

Although the actor in Fig. 4.4 is the Ontology Manager, due to the distributed and integrated nature of the elements that may need to access the ontology, the actor can turn out to be other architecture element, such as the Semantic Assistant. The depicted use cases for this element consist of mostly set and retrieval operations since it will offer other elements information about the system current status and provide support for more complex behaviour.

4.3.2 Semantic Assistance

As referred before, the Semantic Assistant offers semantic reasoning tasks as services to the network. Due to the possible processor intensive nature of the tasks to be performed,

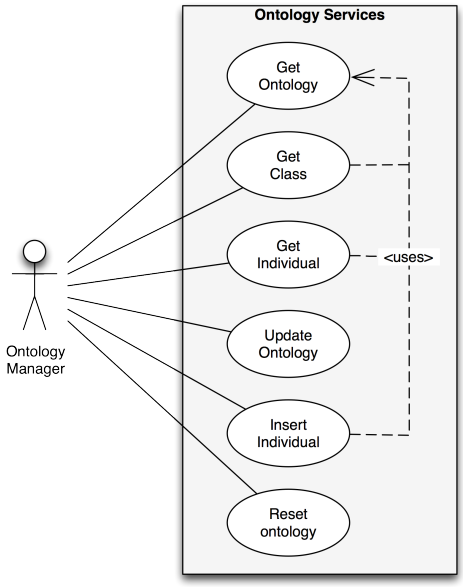


Figure 4.4: Ontology Services – UML use case diagram

this entity can be deployed into on a higher-end device whose services will then be exploited if complex reasoning procedures are required. The services presented in Fig. 4.5 will then be integrated in the processes described on the next sections related to other architecture components. In this UML use case diagram it is possible to catch the major goals of this architecture element – semantically identify a device or service, retrieve the standard set of built-in service for a particular range of devices, check if a semantically equivalent service is currently available and retrieve an existing service translation mapping between two services. To deliver these skills, the semantic assistant heavily relies on the Ontology services to execute the appropriate reasoning tasks.

4.3.3 Discovery and Setup of Devices

When the systems integrator needs to execute a simple check for devices and services currently available the Device Explorer is available to perform a broadcast discovery over the network and it will collect and present to the systems integrator the list of devices found and their hosted services. An extension of this role will also include the metadata values for each detected resource. By having the list of devices and hosted services it is possible to visually check which devices are currently available as well as check for their current metadata values.

An improved version of this feature would be the verification for a mapping within the system ontology for this device instance and retrieve all the information available about it and present to the systems integrator. By adding a sematic tag (commonly a URI) to each available resource in the network it can be used as a unique identifier to

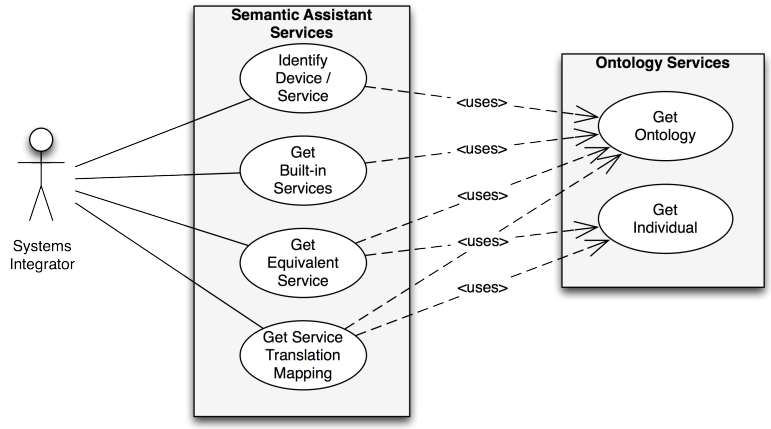


Figure 4.5: Semantic Assistant services – UML use case diagram

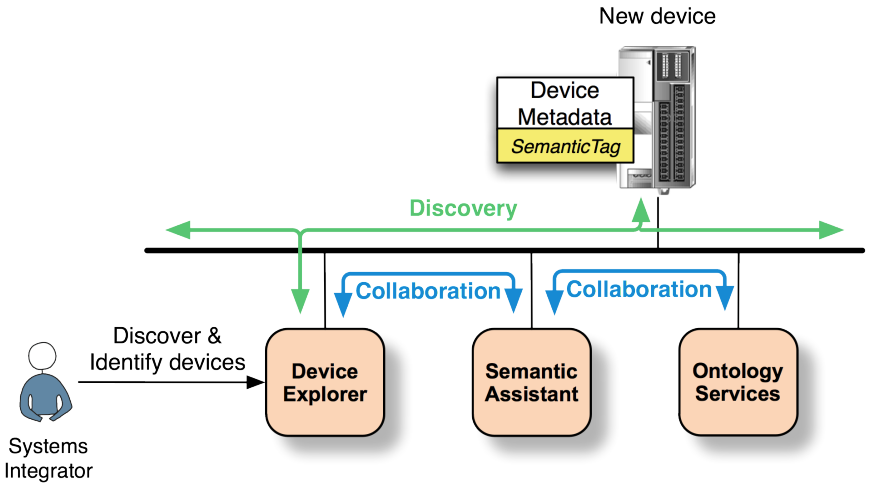
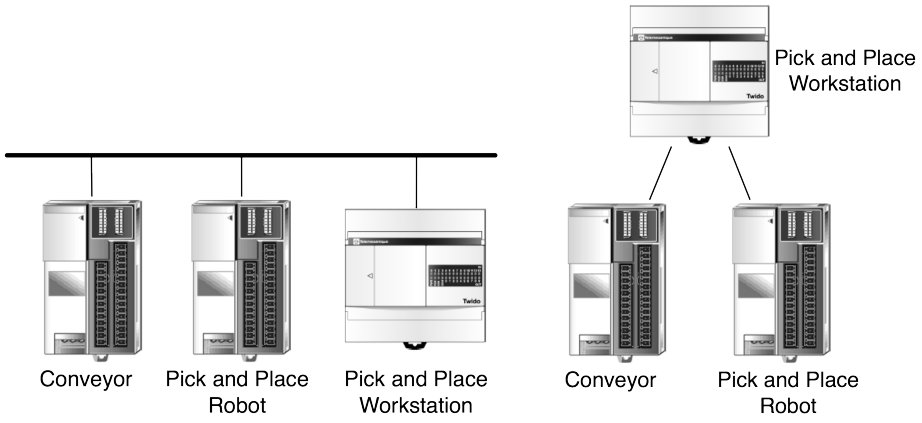


Figure 4.6: Device discovery and identification scenario

indubitably identify it in the context of the current service-oriented application, as outlined in Fig. 4.6. By embedding this kind of distributed discovery mechanism into each device, this simple feature already represents a major input to enhance device out-of-the-box connectivity and setup – one of the biggest known customer demands. During this process there is no need to connect to each device independently using always a different protocol and connector to check its status and configure it.

Based on device and service metadata, it would be also possible to extract not only the physical network topology but also the logical view of the system only based on local information from devices in flat network. This can be presented either by a tree map or by a more complex graphical solution to provide a representation of the different logical levels of an application; e.g. it would be possible to visualize an higher level device with all its sub devices and services under it, and so on. As the example outlined in Fig. 4.7, in



(a) Network view (b) Logical view

Figure 4.7: Device topology visualisation options

a network view of Fig. 4.7(a) all devices are presented at the same level since they are all connected to the same network in a flat layout. In the logical view of Fig. 4.7(b) it is clear to realize that the *Pick and Place Workstation* device is orchestrating both the *Conveyor* and a *Pick and Place Robot* to perform a more complex task.

This can be achieved by agreeing on a common URI format that can be set also using the Device Explorer after discovering a new device in the network by updating the according metadata value. This URI value will determine the logical level of each device and define its position within the current system. Following the previous example, these three devices could include the following URI values in their metadata sets to determine their logical position:

- *Pick and Place Workstation*:
  `http://system.org/equipment/PickPlaceWorkstation`
- *Pick and Place Robot*:
  `http://system.org/equipment/PickPlaceWorkstation/PickPlaceRobot`
- *Conveyor*:
  `http://system.org/equipment/PickPlaceWorkstation/Conveyor`

Some of these parameters values might need to be updated along system lifecycle, so the Device Explorer can directly connect to a found device and change some of the actual metadata values. Also, the Device Explorer can automatically subscribe to the *hearbeating* service of every device found and receive periodic notifications about its liveliness and current status, being warned when a device becomes unavailable or changes to a erroneous state. The systems integrator can also directly poll a particular device status.

The UML use case diagram representing the previous use cases is shown in Fig. 4.8. This diagram demonstrates that the Device Explorer embodies the principal GUI to assist

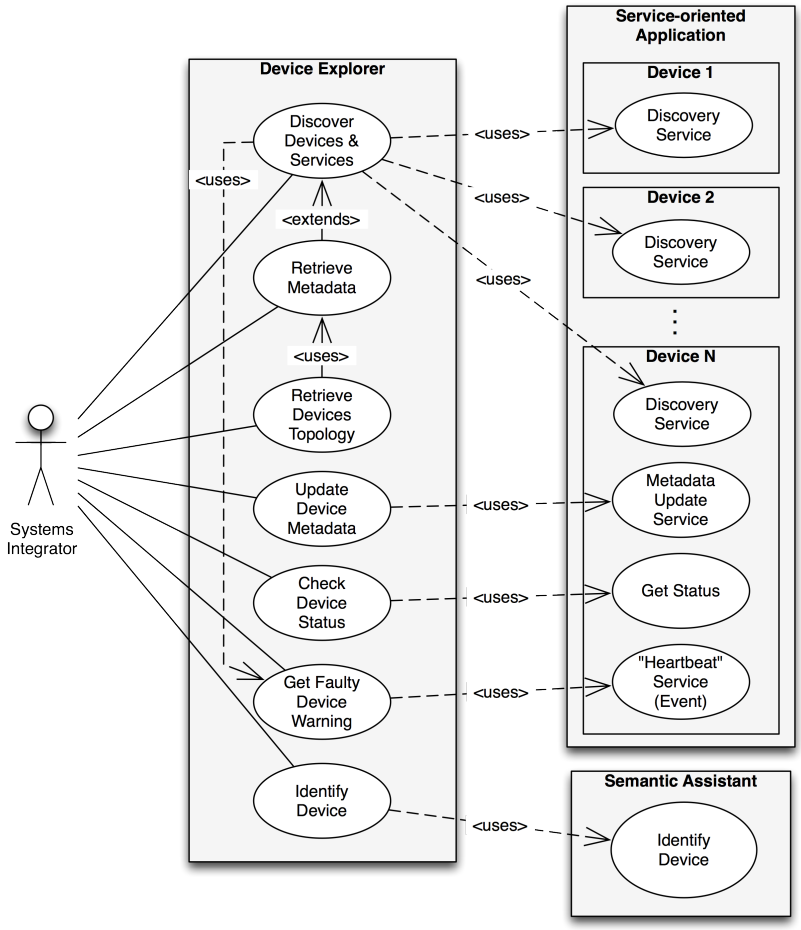


Figure 4.8: Discovery and Setup Devices – UML use case diagram

the systems integrator during early device interaction and employs services provided by the services hosted by the existing devices, but also from other architecture elements such as the Semantic Assistant.

4.3.4 Exchanging devices

The ability to transparently exchange one device by another that exposes the same interface and hosted services to the network is not explicitly related with the proposed architecture itself but with the aspect of being in the context of a service-oriented application. Devices are discovered in the network by their service interfaces and metadata values, i.e. application level interaction. This way when a faulty or deprecated device needs to be replaced by a new one without the need to change the remaining application resources, this new device can be configured to expose the same interface and metadata values as the previous one. The other resources that interacted with the old device will now convey their messages to the new device seamlessly as before. As the example in

Fig. 4.9, a process plan that included a invocation to the web service Y hosted by the replaced device will continue to be valid since the new device implements a service with the exact same interface.

An extension of this approach relates with the application of a Faulty Device Replacement (FDR) routine. For the case when a device is available to substitute a missing one, the human simply needs to retrieve past configuration from Service Repository and use Service Deployer to setup and deploy the appropriate set of resources, as depicted in Fig. 4.10. Of course, depending on system autonomy and safety constraints, this task can be progressively automated. Naturally, this particular case must take into account the type of device and service while checking the implications of substituting that device during system run-time without human intervention.

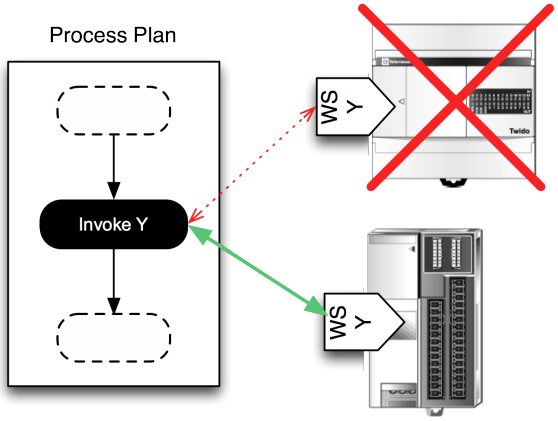


Figure 4.9: Example of exchanging a faulty or deprecated device by another one exposing an identical service interface

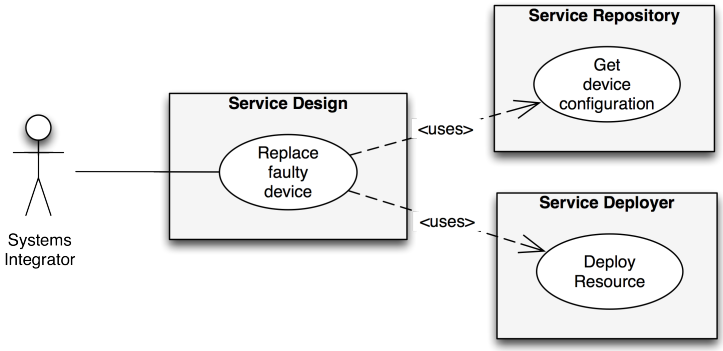


Figure 4.10: Faulty Device Replacement approach – UML use case diagram

4.3.5 Retrieval of built-in services

After having discovered a device in the network it will be possible to map its metadata to the service-oriented system ontology to allow a faster identification, setup and integration of that resource. This will enable the identification and retrieval of the proper set of built-in services available for that particular type of device, which will then facilitate the required interactions to perform the appropriate setup procedure. This process involves the collaboration of the Device Explorer with the Semantic Assistant to determine which set of built-in services to expect from a device from that series. The Semantic Assistant will consult the latest system information by retrieving the up-to-date version of the ontology. With this information in hand it is possible for the Semantic Assistant to process the acquired ontology data and determine which services are available for that particular device and what default parameters values should be set during setup. By having all this information stored in the system ontology it is easier for other elements to follow default setup configurations, requirements and reduce configuration issues. After retrieving the result from the Semantic Assistant, the systems integrator possesses valuable information on how to configure the current device and can use the same Device Explorer HMI to execute this task. The according UML use case diagram is presented in Fig. 4.11.

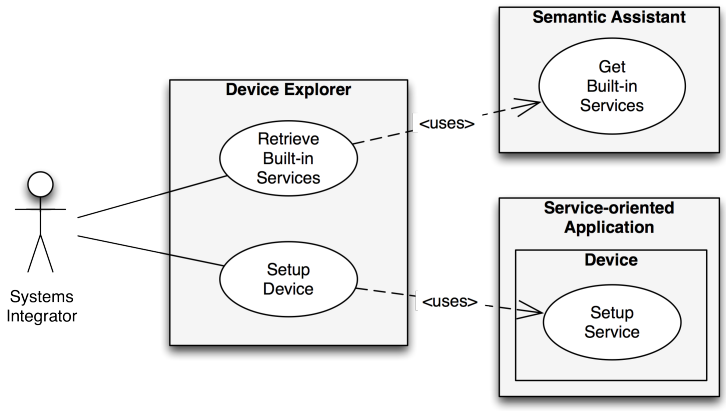
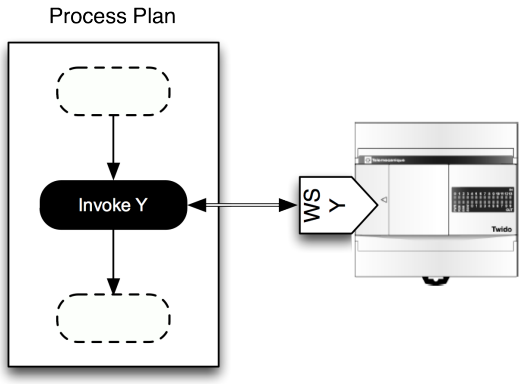


Figure 4.11: Retrieval of Built-in services for a device type – UML use case diagram

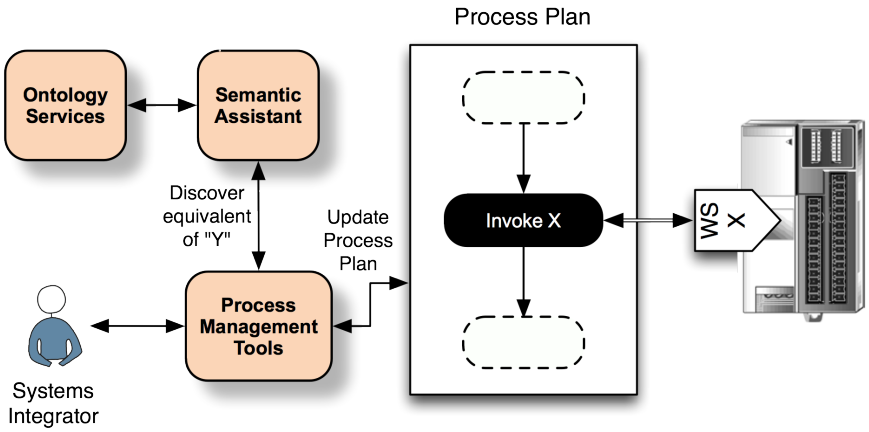
4.3.6 Update of a Process Plan

The existence of a process plan implies that along system lifecycle might be a need to update it to cope with new system requirements, equipment modifications or new process specification and regulations. The proposed architecture takes this matter into account by introducing a Process Management tool to provide an intuitive graphical interface to manage the existing production process plans.

A common issue relates with the need to exchange a device that can have complex



(a) Normal process plan detail



(b) Process of retrieving a service equivalent and updating the existing process plan

Figure 4.12: Updating process plan example

implications on how current process plans are being executed. Also, it might be necessary to replace an existing device by another one that executes the same task activity but presents dissimilar service interfaces. In this case, a process plan that includes a service hosted by the replaced device needs to be updated to use instead another one provided by the new equipment.

As illustrated in Fig. 4.12, a process plan example includes an invocation of the service *WS Y* (Fig. 4.12(a)). After this service being unavailable, there is a need to map this activity to an equivalent service. The systems integrator will be aided by the Semantic Assistant to retrieve the fittest service to replace the missing one (*WS X* in Fig. 4.12(b)) and update the current process plan, if a service in these conditions exists in the moment.

This use case role falls in the context of the integration of particular OEM equipment where the systems integrator is not allowed to modify the actual service interface. Since the new equipment is still compatible with a service-oriented approach, there is no need to apply the software-wrapper solution commonly applied to legacy equipment.

Besides these agility enhancement skills, a Process Management tool should also allow systems integrator to create his new process plan and preview it before putting it in action. Consequently, the Process Management tool allows the launching of an instance of an existing process plan by relying on an embedded (or possibly distributed) process executor. The process executor is responsible for running and supporting the monitoring of every process currently being executed in the system. Once again, due to the distributed nature of the architecture, it would be possible to replicate this functionality across different machines relying on the abstraction of service interfaces.

Before running any process instance there is a need to check if the required resources are currently available in the system. If a resource becomes offline in the middle of a running process, the process monitor can trigger a resource replacement automatically or over user validation, depending on system definitions, by relying once again on the Semantic Assistant services.

The UML use case diagram including the use cases previously explained is shown in Fig. 4.13.

4.3.7 Semantic gateway

The presence of semantic gateways can provide an important input, particularly in moments on transitions, such as when a machine fails or it is down due to a maintenance intervention. It is then important to provide an immediate solution, even if compromising the performance of that particular system segment. Semantic gateways can provide immediate replacement functions by executing a translation procedure to handle the flow of information in between two or more semantically distinct entities.

After identifying the missing or faulty devices, the Semantic Assistant is solicited to retrieve in the network services equivalent to those currently unavailable. While discovery and identification implies service description and semantic tags comparison, the interaction might require a translation mechanism to adapt to a possible unlike interface – services coming from different sources might differ in the interface although implementing the same exact essential function.

The Semantic Assistant can extract translation guidelines from the system ontology to determine the required translation mappings to be implemented by the gateway service. After gathering these translations guidelines it will be possible to automatically create a service possible of being deployed into a free device using the Service Deployer. This translator service will emulate, i.e. copy the interface of the missing service and translate any invocation that it receives to the new service interface that will temporarily replace the original service. For the device that was using the now inexistent service, it will discover the translator as if it were the original service and start to use it. See Fig. 4.14 for an example and Fig. 4.15 for the according UML use case diagram.

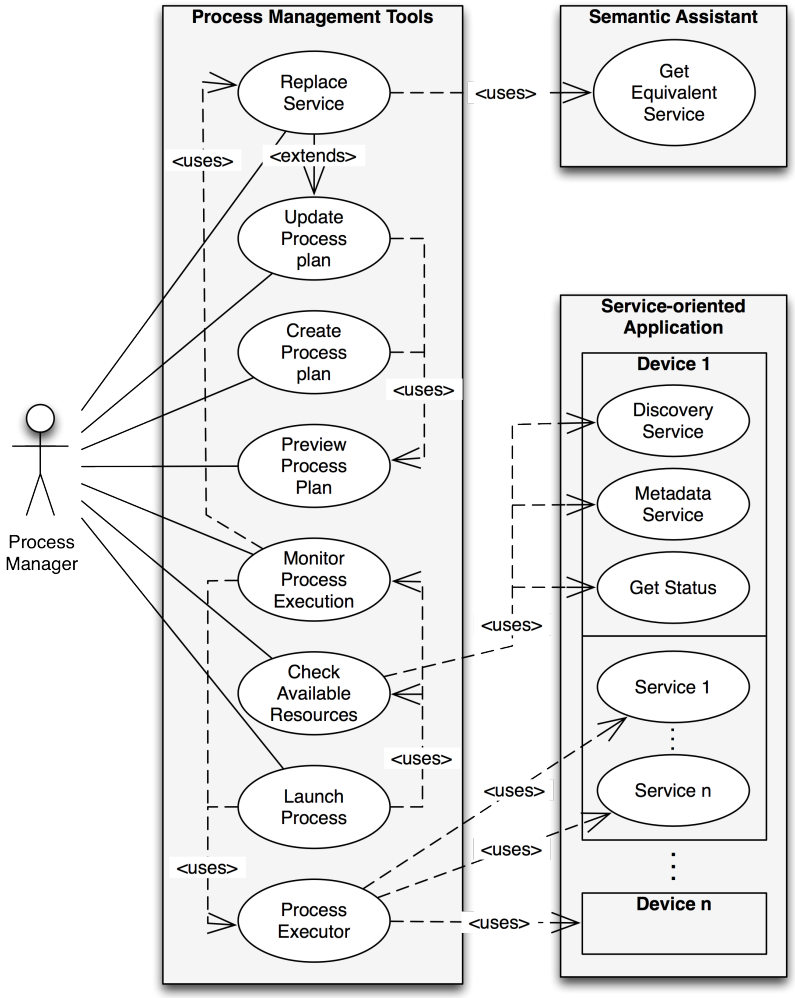


Figure 4.13: Replacement of a service from an existing process plan – UML use case diagram

4.4 Wrap-up

The current architecture proposal introduces a set of elements to support a more agile, transparent and effortless lifecycle support to the device level in service-oriented industrial automation installations. This collection of service-oriented elements can compose a mouldable infrastructure focusing to ease common integration aspects, such as device discovery, identification, setup and process modification to cope with an unexpected event.

The ability to reconfigure system devices and process plans is improved when comparing with more traditional approaches in terms of interoperability and hardware abstraction targets while remaining compliant with domain know-how. By laying over open web standards and focusing on interoperability, modularity, uncomplicated management and adaptation, particularly enhanced by the use of the deployment built-in

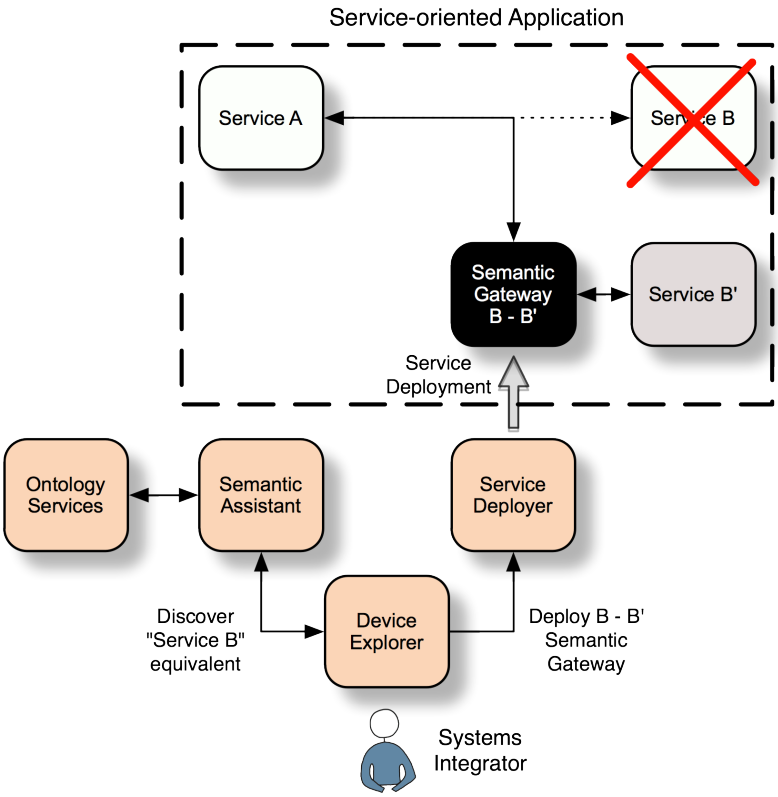


Figure 4.14: Example of the deployment of a semantic gateway

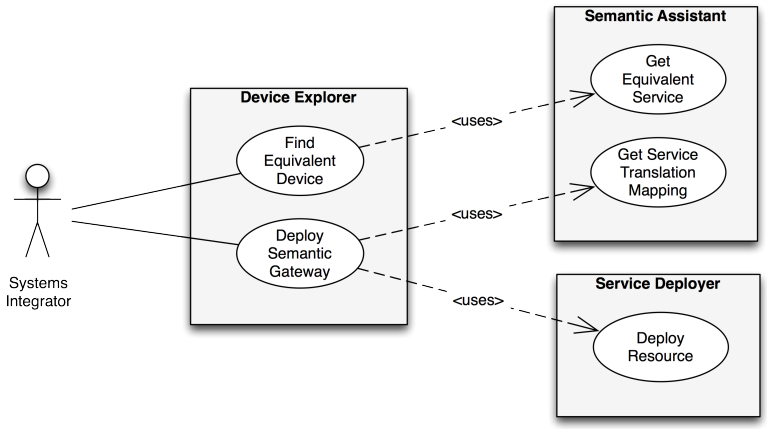


Figure 4.15: Deployment of a semantic gateway – UML use case diagram

service, it is possible to define a set of components that together form a customizable support infrastructure. This infrastructure is modular and adaptive enough to evolve along with system specificity and requirements during its lifecycle. Each architecture element exposes its services in the network, which will enable a customized composition of modules and a mutual transparent interoperability. The ability to chose the fittest combination of elements to be set for a particular installation is a major outcome since not only it permits the customization of the end-user infrastructure but also allow equipment and solution providers to present a product offer with distinct levels of quality of service.

Due to the distributed and interoperable nature of the proposed architecture, the elements can be deployed in multiple combinations of hosting devices. It can range from a single device deployment (centralized in a single equipment instance) to a totally distributed ecosystem where each element lays on a different physical device chosen to optimal performance taking into account the element requirements and tasks to be performed by it. Also, since each element is interfaced by a service, it would be possible to replace or update an existing component without extensively affecting the remaining infrastructure. In this case, only those elements that depend on it will be affected if no redundancy measures are put in place. These measures may include replication of elements exposing the same interface or a system that automatically deploys a new element instance if the previous one becomes unavailable.

At the limit, the vision beneath the goal of the proposed architecture can itself be applied to definition of the device lifecycle support infrastructure this time in the scope of a solution provider. In this case, the solution provider could create his own customized solution for a particular client by deploying the architecture elements that best fit client needs in transparent and effortless manner as the present architecture depicts for the creation of service-oriented applications addressed to industrial automation domain. The solution provider could also rely on ontology and semantic assistants to assist during the design and deployment process, being then possible to monitor and diagnose the deployed solution. This extension of the proposed solution is not pursued in the context of this thesis and can be considered for possible future research.

When comparing with more traditional approaches, this solution may require additional processor and memory requirements, while introducing concepts unfamiliar and sometimes disputed in the industrial device level domain. Training and clarification sessions are then fundamental to overcome these issues.

Another aspect relates with the security concerns that must be taken into account particularly when accessing and modifying devices configuration and ontology without compromising overall system integrity and performance. The access to the most critical system elements should be protected from external interventions being necessary to deploy authentication and authorization solutions, and even encryption when exchanging critical proprietary data.

5

Service-oriented Device Model

5.1 Introduction

The current section proposes a device model compliant with what was defined in the reference architecture in order to support the expected levels of agility during device lifecycle support.

In this context, a device is to be seen as the main logical entity that abstracts an element of the actual application, while the services it hosts represent the functionalities or tasks that a particular element can execute. These services will allow others to exploit them while pursuing their own goals (see Fig. 5.1). This methodology also supports a more collaborative approach where a device hosts a set of services consistent with its role on the system and, at the same time, it exploits services hosted by neighbour devices to accomplish its behavioral instructions. In slight opposition to original SOA principles where services are the principal building blocks, when adapting these concepts to to industrial automation device level the previous modelling was preferred regarding the device- or equipment-centric vision in the domain. Nevertheless, it must be noted that a device is simply a modelling abstraction using a predefined service interface.

Both devices and services can be discovered, identified and employed to execute a predetermined assignment or be composed to create more complex entities. Moreover, it is possible to distinguish between logical and physical devices. In this context, the physical device concept refers to the physical apparatus comprising casing, physical sensors and actuators, processor, memory, communication interfaces and firmware, i.e. the device itself as a real-world physical entity, such as a PLC, an I/O device or a PC. The definition of logical device refers to a software unit that is used to abstract a particular system element not explicitly associated to a physical entity but from the application logic point-of-view. The application can then built upon several logical devices and according

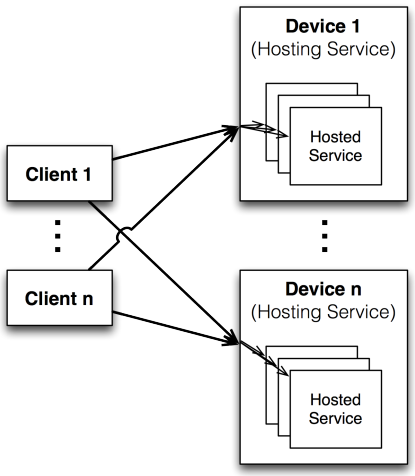


Figure 5.1: Device and hosted services overview

hosted services that interact between them.

5.2 Device Model

The device model shown in Fig. 5.2 represents the physical device as a repository of logical devices and corresponding hosted services; also referred as device platform.

As a short summary, the physical device is abstracted by a logical device (*LD**). This logical device already includes several built-in generic services (*bgS*) that will deliver capabilities and features considered standard for a particular range of devices from the moment the device is taken out of the box and connected to the system. This set of built-in generic services includes a deployment service that will allow the download and management of new services to the current physical device. These services can consist of deployed generic services (*dgS*), which can even augment the level of embedded complexity and interoperability to cope with more particular system requirements. Concerning the user application, it would be possible to deploy logical devices (*LD*) to abstract application components and their user services (*uS*), which as whole are expected to implement the device behaviour, and more significantly, the expected role in the current application context.

It is then essential to further detail the different kind of resources that compose the proposed device model.

5.2.1 Generic Services

As introduced before, the physical device is expected to already embed some built-in services that will allow the deployment of user applications but also other essential services to improve device accessibility and accountability. These services can ease the setup, management, monitoring, and diagnosis tasks from the systems integrator point-of-view

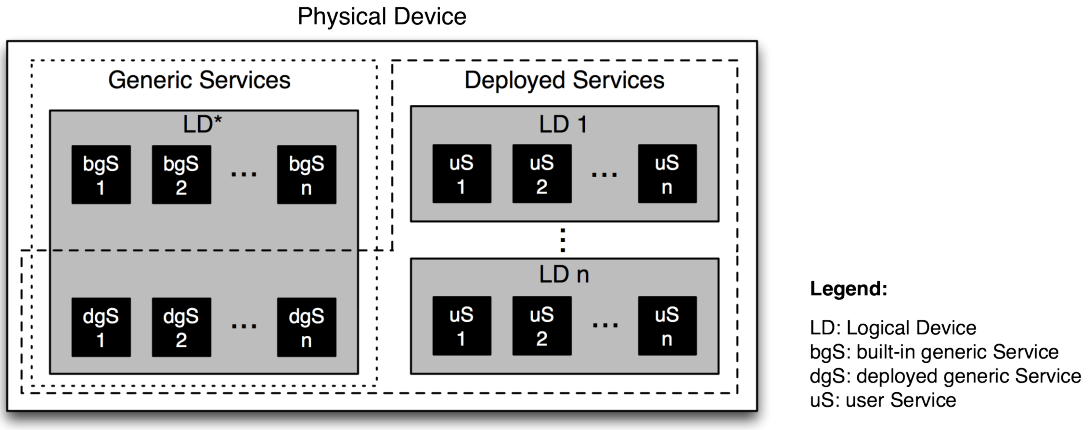


Figure 5.2: Device model overview

or cope with more particular or critical system requirements.

These services are deployed by the device builder during its production phase and are immediately accessible when taking a new device out from the box at the end-user site. This approach follows a mechatronics vision where a physical device besides comprising the electronics and often mechanical parts, it also includes a service-oriented firmware including embedded features and services running on top of a IP-based communication interface.

These built-in generic services cannot be removed or modified by the end-user – if the systems integrator needs to add new services, he can do it through the deployment service also embedded. The deployment service is itself a built-in service, which will allow the systems integrator to deploy its own resources i.e. logical devices and its hosted services.

For each type of device, a collection of generic services can be specified encompassing the standard requirements and purposes for that particular series of devices. This set of services can also be divided into mandatory and optional subsets. The mandatory set is deployed into a device by its builder during production, while the optional services can be deployed when needed. This approach allows a more precise customization of devices to face particular client requirements.

Although the device builder might want to develop its own proprietary services to distinguish its offer from its competitors, the communication technology is expected to remain open and standard. This approach will increase device interoperability with other devices and integration tools. The services interfaces and documentation can then be shared in order to let the end-user freely choose how to employ them and which tools to use.

These proprietary services should provide complex services to deliver a more sophisticated level of information than simply get and set data values. Proprietary services should provide more intelligent responses (higher level) to the client based on available

information processing. For example, instead of simply getting a collection of maintenance data values, a maintenance service can provide a summary of activity focusing the most common points of interest, still based on those parameters but executing some processing before sending it to the client. At the same time, this approach is expected to open new business models by adapting the level of embedded added value following client requirements.

Finally, this set of generic services facilitate incorporating and managing services (and applications) into devices. Subsequently, the lifetime of the service is a subset of the lifetime of its host: the device. These generic services ease the adaptation of each device to different scenarios of execution by improving the agility and openness of the process of installing and managing new components.

5.2.2 Deployed Services

The current approach implies the existence of a dynamic deployment service built-in in each physical device. This way it would be possible to customize it by deploying new resources: logical devices and their hosted services.

The user application will be designed and deployed in a form of logical devices. These logical devices will represent the logical entities that can be observed or inferred from the current application, exposing their hosted services as their capabilities that other resources can make use of.

An application can even compose several of these logical devices into several layers of increasing abstraction, in an orchestration or choreography manner – application construction based on existing building blocks. The interaction between these is supported by their hosted services where some of these logical devices not only host services but are also clients of services hosted by other logical devices.

Still, it would be possible to deploy services that will enhance the functionality already provided by the current built-in generic services – these services are also considered generic to that particular range of devices, although they are not considered mandatory to the majority of applications. Some application functionalities can be considered generic enough and be reused across several applications, being deployed whenever they are required. For example, it would be possible to deploy into a PLC device some services that can control some common system components, such as conveyor belt, a pallet buffer or a RFID tag reader. This approach will avoid the need to recode those components every time they are needed and still remain consistent with previous implementations or development guidelines.

5.2.3 Built-in Services

Built-in services, by definition, are considered generic for a particular category of devices and offer an interoperability channel to access, manage, monitor and diagnose the the actual physical device. Being part of the original firmware, they cannot be modified by

the user, although it is possible to extend the original set by deploying new resources.

5.2.3.1 Dynamic deployment Service

In the context of a SOA for devices, the dynamic deployment service is the instrument that will allow more customizable, and subsequently more agile systems. This is also achieved by promoting a clear independence between the development of the device platform and the development of the application services.

The process of deploying services is performed in two steps:

- The service implementation code is first uploaded on the device platform through a service operation: depending on the technologies used by the device platform, the service implementation may be usable immediately (e.g. when using interpreted code or some dynamic code loading and linking mechanism) or may require a reboot (e.g. when the service implementation must be integrated with the platform firmware).

- The device is configured by creating a new service instance based on the previously uploaded service implementation. The specification includes a detailed description of the device and service metadata which altogether will define the actual service description interface.

To better understand the elements that compose the device model and relations between them, the concept of resource is now defined.

5.2.3.1.1 Resources

In the context of this work, a resource consists of an entity possible to be managed through a software layer interface. This resource can be a logical entity or abstract a physical object, such as a PLC, a I/O device or a PC.

Besides the device and service resources already enunciated, the proposed model requires a new resource:

- *ServiceClass*: this resource is used to describe the service implementation, which contains both generic information that must be provided by all implementation types, and implementation-specific information, in particular the service implementation code and initialization parameters.

The relationship between the different types of resources included in the proposed device are presented in the UML class diagram in Fig. 5.3. From this diagram it is possible to apprehend that all device model elements inherit from the *Resource* class. Every resource includes a *ResourceURI* which refers to the URI of the resource class representation or instance representation and a *SelectorSet* that identifies the resource instance to be accessed if more that one instance is available.

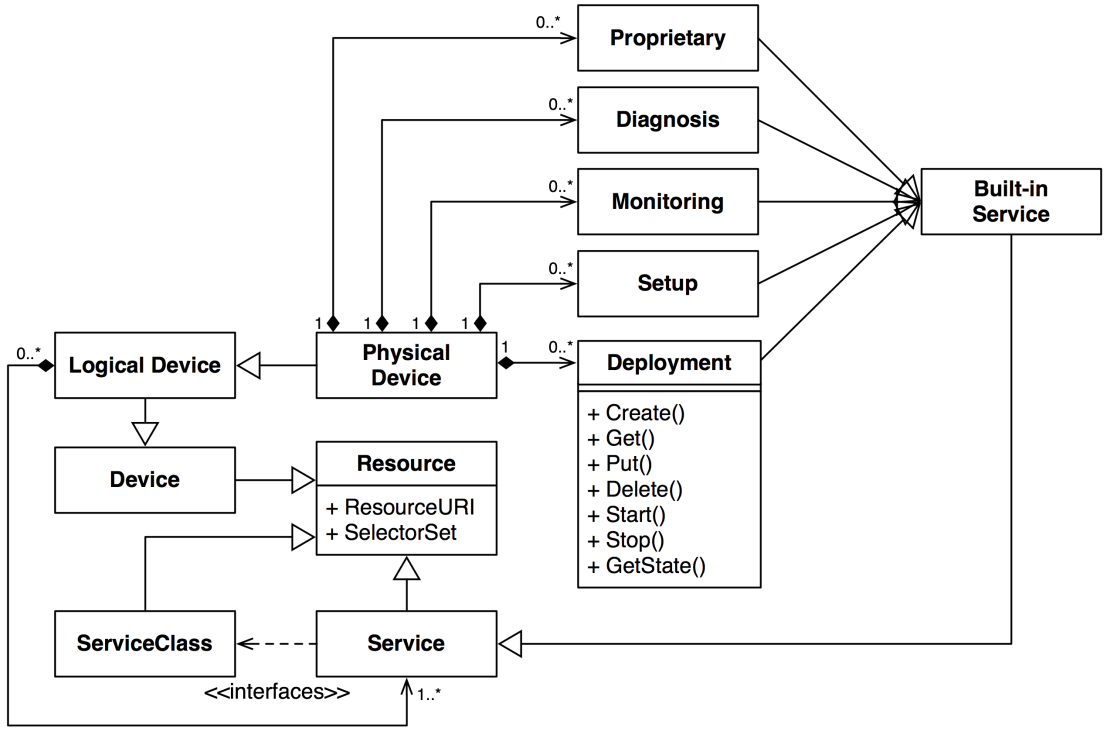


Figure 5.3: Device model resources – UML class diagram

It is possible to manage these resources in a service-oriented manner by using the proposed *Deployment* built-in service operations. This set of operations can be used to manage every *Device*, *Service* and *ServiceClass* instances in the context of physical device.

As said before, a *Physical Device* is a particular case of *Logical Device* since it relates to the real device that will host all other resources. Moreover, the *Physical Device* will include a collection of built-in services to handle setup, monitoring, diagnosis, deployment of new resources and it is open to be extended by proprietary services that the device builder might find competitive or essential to make it available.

More details on resource operations and lifecycle are present in a subsequent section. Some details on service operations, such as input and output parameters, were omitted in Fig. 5.3 to improve readability.

5.2.3.1.2 Resources Lifecycle

It is important to distinguish the two phases of the lifecycle of a resource: the deployment phase and the activation phase. The deployment phase is usually performed in two steps:

1. The service implementation (*ServiceClass*) is first uploaded on the device platform, i.e. into the real device using the corresponding *Deployment* service operation.

2. The *Device*, or more concretely a *Logical Device*, is configured by registering one or more *Service* instances based on the previously uploaded service implementations as its hosted services.

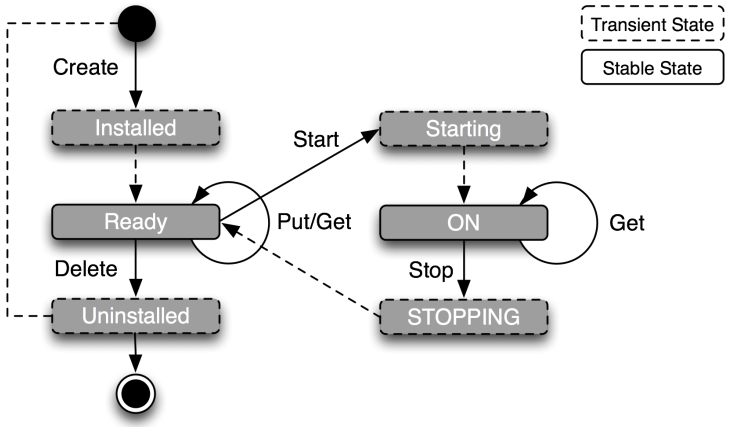


Figure 5.4: Device and Service resources – UML state machine diagram

A *Service* interfaces a *ServiceClass* instance previously deployed on the physical device. In summary, a services consist of descriptions of *ServiceClass* instances that embed the actual implementation – it is possible this way to ensure the independence between the actual implementation technology and the service interface.

The lifecycle of the resources follow the proposed state model presented in the next section.

5.2.3.1.3 Resources State Model

Although these three resources: *Device*, *Service* and *ServiceClass* have their own lifecycle and state, their lifecycle is strongly linked. All these resources have a common parameter that stores its current state. This feature allows the integrator to retrieve the current resource state at any time, being, at the same time, the major parameter that can condition the resource lifecycle state-machine in order to detect possible lock situations in interdependent resources.

5.2.3.1.3.1 Device and Service Resources states

Both *Device* and a *Service* instances can be managed and monitored during its lifecycle by using the *Deployment* service operations, as presented in Fig. 5.4.

An instance of one of these resources can then be in one of the following states:

- INSTALLED – The resource has been successfully installed.
- READY – All references that the resource needs at deployment time are available. This state indicates that the resource is either ready to be started or has stopped.
- STARTING – The resource is being started, the *Start* operation has been executed but is not yet completed.

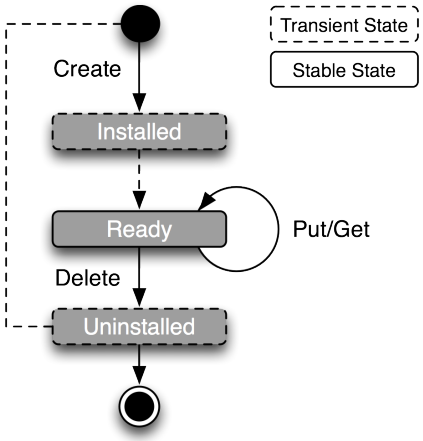


Figure 5.5: *ServiceClass* resource – UML state machine diagram

- ON – The resource has successfully started and is running. This means that the resource (*Service* or *Device*) is now possible of being discovered and employed as part of the current physical device.
- STOPPING – The resource is being stopped. The *Stop* operation has been executed but it is not yet completed.
- UNINSTALLED – The resource has been uninstalled. It cannot move into another state. All resources references and files are removed from the physical device.

In summary, a *Service* or *Device* resources are created and stored in the persistent storage of the device platform, being then initialized to become available for use within the context of the current application. Also, it is possible to turn off that same resource when it is not needed for the new context, or even remove it to reduce the memory footprint of the device platform.

5.2.3.1.3.2 ServiceClass resource states

A *ServiceClass*, as said before, refers to the concrete service implementation, which contains both generic information and implementation-specific configuration and program.

Comparing with previous *Device* and *Service* lifecycle states (Fig. 5.4), the *ServiceClass* resource only includes the following possible states: INSTALLED, READY and UNISTALLED. These have the same meaning as in previous scenario.

Since the *ServiceClass*, in general terms, only refers to the implementation part there is only a need to store and remove it from the device platform. The proposed *ServiceClass* state diagram is depicted in Fig. 5.5.

5.2.3.1.4 Operations on Resources

Each resource can be managed through the available set of operations that allow the systems integrator to control their lifecycle in the scope of the physical device that hosts it and actual system context.

All operations were designed to follow the common request-response MEP. The following operations can be performed for every instances of classes inherited from the *Resource* class:

- *Create*: creates the resource instance.
- *Get*: retrieves the resource representations and parameters values.
- *Put*: updates the resource representations and parameters values (can only be executed when resource is in READY state).
- *Delete*: deletes a resource instance.
- *GetState*: this operation gets the current state of the resource identified by its according *ResourceURI* and *SelectorSet* values.

Besides the previous set of operations, each *Device* and *Service* instance can also be managed using these extra two operations:

- *Start*: starts the resource making it discoverable and possible to be invoked.
- *Stop*: this operation stops the resource. It hides the resource from the network, although it is still stored on the device platform (READY state)

5.2.3.1.5 Resources Interdependencies

The logical interdependencies between the different types of resources will sometimes constrain the execution of the previous operations.

The deployment of resources is always done using *Create* operation included in the *Deployment* built-in service which will return a unique resource identifier (*ResourceURI*) in case of successful execution. Using this identifier as resource selector it will be then possible to retrieve its current state by invoking the *GetState* operation.

This operation will create a new element in the device configuration file that stores the state and content of each resource deployed into that particular device. This configuration file can consist of a XML file available within the device firmware. This file is updated in accordance with the activity associated to deployment service operations.

The deployment of a *Service* instance will simply consist on the registry of a service class to a particular device element – that service will then be available from that device as an hosted service. This operation will be also constrained by the resources already available on the current physical device. It will not be possible to deploy identical logical devices into the same physical device, even if each one have its own *ResourceURI*, since

it will be hard to distinguish between two or more identical devices during discovery phase for example.

A *Start* operation can only be applied to *Device* and *Service* resources, since a *ServiceClass* resource only consists of a service representation and implementation details. A *Start* made to a service from a logical device not currently in ON state, will imply that the service remains in STARTING state until the device gets into ON state. The service can only be discovered and invoked while being on ON state, as well as the device that exposes that service. A *Start* made to a device will imply an automatic subsequent *Start* of all the services it hosts. A single *ServiceClass* instance can be registered to several different devices, either hosted on the same or on two or more devices. Nevertheless, in the last case, an identical *ServiceClass* instance must be also deployed on those devices.

The *Stop* operation, for the same reasons as for *Start* operation, cannot be applied to *ServiceClass* resources. When a hosted service is stopped, it remains registered in its device, but it will not be available to be discovered or invoked since it returned to READY state. Still, to permanently suppress a hosted service from a device, the operation *Delete* must be used instead. As for the *Start* operation, a *Stop* operation over a particular device will imply the stoppage of all its hosted services. Contrary to *Stop* operation, the *Delete* operation will definitely suppress the resource from the device platform, being necessary to redeploy it later if necessary. Again, the relations between different resources are sometimes critical and should be taken in account while performing these resource management tasks.

The *Delete* of a *ServiceClass* instance is always constrained by the hosted services that are still using it. The operation will only be successful if that particular instance is not registered in any of the logical devices as a hosted service; otherwise the service will return the list of devices that are still registered to this instance.

The result of a *Delete* operation over a *Service* registered to a *Logical Device*, or hosted service, will imply that the device that previously hosted that service will no longer expose it, even after a restart of device, except if a new deployment is done.

As the previous operations, a *Delete* operation over a device will also imply the removal of all its hosted services. Still, interconnected *ServiceClass* instances will remain untouched, since other devices might be using them as *Service* instances implementations.

5.2.3.1.6 Persistence

Persistence is fundamental to avoid reconfiguring or redeploying resources into a device each time it is powered off and on again. The middleware must guarantee the management of resources states along their complete lifecycle, as well as avoid the modification of the built-in services deployed by the device manufacturer. Not only the configuration parameters must be kept, such as network addressing, access parameters, diagnosis and monitoring history, but also the application elements that were deployed and active within that physical device before the shut down.

Regarding the set of built-in services, the implementation will remain proprietary and it will depend of each device characteristics. This way, although the interface of these services can remain unchanged, the way to implement persistence features for these kinds of services is out of the scope of this work.

The elements deployed in a physical device are saved in the device configuration file that will be permanently stored on the device file system. Any update of configuration or resource deployment will alter this same file. This way, every time a device is shut down this file will be kept in internal persistent memory. Every time the physical device is powered on again, the device configuration file will be interpreted and the latest active configuration will be executed.

5.2.3.2 Setup Service

In the context of this work, two different aspects of the setup phase were taken into account when projecting the setup service. Initially, when plugging a new device into the shop floor network there is a need to discover and identify it in the context of the ICT infrastructure. A posterior phase relates with the need to configure the device to be in accordance with particular network definitions.

5.2.3.2.1 Discovery

If needed in current service-oriented application context, the device is the discoverable entity in the network, being possible to be retrieved through the services it hosts. The chosen firmware must enable the discovery of all devices on the local network, and establish a dialog with a device without having to know its current IP address – transparent connectivity.

Types and *Scopes* are used as filtering criteria during the discovery process.

- *Type*: Abstract or functional types describing the device, e.g. PLC, I/O device, Industrial PC, etc.

- *Scope*: Optional parameter that can be defined to categorize application-defined information used to identify the device, such as its location or position on the device network topology. Several scopes can be used for a single device.

The device *Type* can be defined in the product catalogue, so that the user can easily discover the device he is looking for and avoid all the others present in the same network. Systems integrator can modify the scope in order to customize each device according to actual application scenario.

After discovering it, it is possible to request its metadata and retrieve all the parameters values needed to precisely identify that particular device. The proposed basic set of metadata parameters is enumerated in Table 5.1. The *FriendlyName* value can be customized to address the needs to easily identify the device in the network by the systems

Metadata Parameter	Type
Manufacturer	String
ManufacturerUrl	URI
ModelName	String
ModelNumber	String
ModelUrl	URI
PresentationUrl	URI
FirmwareVersion	String
FriendlyName	String
SerialNumber	String
DeviceStatus	Enum
SemanticTag	URI

Table 5.1: Basic set of device metadata parameters

integrator, while the *DeviceStatus* value indicates the current status of the physical device, following the proposed resource state model. The retrieval of the device status is an immediate requested feature to allow any time to check device availability. The *SemanticTag* value store a URI to be used during semantic identification and to ease reasoning tasks. The other parameters are self-explanatory and should be filled with the appropriate catalogue parameters. Nevertheless, it is would be also possible to extend these if new requirements are introduced.

5.2.3.2.2 Configuration

For TCP/IP Ethernet-capable devices, the setup phase has a major importance since it is the first step to allow the communication with a new device. One of the most common problems is not knowing what is the default IP address of a new device to be able to access it and change it to be in accordance with actual network parameters. Although the service communication layer abstracts these and promotes interactions based on the services each entity offers to the environment, most of the times there is still a need to define these parameters in accordance with the remaining network definitions. By allowing the retrieval and set of the device IP address in a standardized way, the integration phase is simplified and open to standard tools. The same approach can be applied to other configuration parameters. Almost all these services can be considered bidirectional. Bidirectional in this context means that it is both possible to retrieve or set the configuration data.

The process of configuring a device can be realized by exploiting *set* and *get* operations over variables that compose the configuration data. This configuration data must follow a common cross-device data model to improve the relevance of employing this generic methodology. The resource description model should be designed to be extensible to cope with particular domain, application needs and available standards. The innovation does not reside in the identification of configuration data, but rather on the way to leverage SOA for doing it.

Operation	Parameters	Data	Type	Access
Device Metadata	Metadata Values	FriendlyName	String	R/W
		Type	String	R/W
		Scope	String[ ]	R/W
		SemanticTag	URI	R/W
IP Config	IP Address	Address1	String	R/W
		Address2	String	R/W
		Address3	String	R/W
	Parameters	ReservationTime	Long	R/W
		HoldupTime	Long	R/W
		LinkFailureMode	Enum	R/W
Password Management	Change Password	Login	String	WO
		Old password	String	WO
		New password	String	WO
Serial Port setup	Configuration	Baud rate	Int	R/W
		Transmission mode	Enum	R/W
		Flow control	Enum	R/W
		Parity	Enum	R/W
		Data bits	Int	R/W
		Stop bits	Int	R/W
		Number of retries	Int	R/W
		Responses timeout	Int	R/W
		Enable broadcast	Bool	R/W

Table 5.2: Example of device configuration operations and data

Concerning device *FriendlyName*, *Type* and *Scope* and *SemanticTag* it is possible to update of these during the device lifecycle. Table 5.2 presents an example of device configuration operations and related data types. This table includes operations for modifying values related to the device metadata, IP address and other network configurations, password management, and for the given example, an extra operation to setup the Serial Port parameters.

The set and retrieve of configuration parameters should comply with the defined configuration model for the current range of devices adding optional parameters related to particular features or communication interfaces such as a Serial Port, Modbus server or legacy Simple Network Management Protocol (SNMP) definitions.

5.2.3.3 Diagnosis Services

The objective of having a service responsible for the device diagnosis relates with the fact that systems are configured and programmed to be predictable and reliable. However, devices fail and there is a strong need for continuous monitoring and diagnosis of events. This way, each device must be easily reachable to retrieve all necessary diagnostic information to better determine the real cause of the fault, alongside asynchronously notifying the system whenever there is an internal or related event to consider. These event sources have the objective to notify its subscribers about the possible reasons or sources of errors, faults, or detected abnormal behaviours.

These events can then be subscribed and collected by intelligent aggregators, which process them having a broader vision over the system to determine the overall system

Operation	Parameters	Data	Type	Access
Diagnosis Notifications (Eventing)	Error	Code	String	RO
		Timestamp	String	RO
		Description	String	RO
	Warning	Code	String	RO
		Timestamp	String	RO
		Description	String	RO
Counters Reset	-	-	-	Cmd
Ethernet port statistics	Transmit statistics	Frames OK	Long	RO
		Collisions	Long	RO
		Excessive Collisions	Long	RO
		Carrier sense errors	Long	RO
		Internal MAC errors	Long	RO
	Stats Resume	Content	String	RO

Table 5.3: Example of device diagnosis operations and data

issue, causes and origin.

Table 5.3 presents an example of device diagnosis operations and related data types. This set of operations may include operations to retrieve device statistics, such as network and TCP/IP statistics. Each device also triggers diagnosis events for errors or warnings that include the according code, timestamp and brief description. All counters and history data can also be reset through the according service operation invocation.

5.2.3.4 Monitoring Services

As for diagnosis, monitoring services serve as a door to access information relevant to ensure a constant monitoring of each device and the system as a whole. The monitoring information at physical device level can comprise information related to network statistics, current status, fault historic data, etc. Basically, this sub-set of services is dealing with Feature-based Monitoring indexes, i.e., monitoring information generated by the hardware or software associated to the device.

The focus should be put over the ability to retrieve monitoring data, preferably in a pre-processed format, but also over the asynchronous ability to push notifications related to a *Heartbeating* feature and events related to the normal device behaviour, such as the successful termination of an executed task. An example of application is demonstrated in Table 5.4, which presents an example of device monitoring operations, events and related data.

5.2.3.5 Proprietary Services

The proposed device model allows the extension of the basic set of built-in services by allowing the deployment of extra proprietary services. This feature is envisioned to be exploited not only by the original device manufacturer but also by OEM companies that can build their own proprietary services implementations upon it before selling the complete solution or equipment package. The device manufacturer can create a new line within each range of devices to exclusively address this branch of users.

Operation	Parameters	Data	Type	Access
Heartbeat (Eventing)	DeviceStatus	Value	String	RO
		Timestamp	String	RO
Execution Event (Eventing)	Event data	ID	String	RO
		Description	String	RO
		Timestamp	String	RO
Activity Summary	Activity data	ID	String	RO
		Timestamp	String	RO
		Summary	String	RO
Counters Reset	-	-	-	Cmd
Set Time Interval	Time Interval	Value	Int	R/W

Table 5.4: Example of device monitoring operations and data

It can be envisaged for an OEM to sell its own products interfaced by devices that already embed their own collection of built-in services, particularly to handle the management, monitoring and diagnosis of the application itself. These basic services can be seen as the basic packet. Then, the client can request extra functionalities concerning more advanced services in comparison with those initially offered, but also application services that can be use control a particular element of the system.

This approach enhances OEM agility to handle customized demands from its clients, embedding at the same time an important added-value to its final product. It would be easier for an OEM to compose its solution based on pre-built building blocks – these resources that compose the complete solution will be stored in the device as built-in services. Although the end-user cannot change these a priori, it is possible to add new ones using dynamic deployment service, if the OEM wants to add them to its product. These building blocks and functions, i.e. logical devices and services, can then be reused in a new application. It is then highly advantageous to have generic services to control and manage common system parts that can be replicated in a wide range of implementations. Subsequently, the cost to develop new solutions will decrease at the same time as time-to-market.

In industrial automation, there are certain elements and functions that are frequently required for a particular domain of application. These functionalities can be abstracted and made available as logical devices that can be deployed into service-oriented automation devices through the dynamic deployment service. Due to the abstraction possible with this approach, the actual software components can cover any domain of application, from field devices to high level business servers. The device vendor can provide a complete collection of services ready to be deployed to handle common implementation functions. It is then important to enforce a strong interface design to promote reusability, clearness and ease of access to gain client attention to apply these pre-developed services.

5.3 General-purpose services vs. classical Management services

Since the present work also addresses the application of SOA vision to industrial automation device level, it is important to discuss the different ways to access and manage a device through service interfaces. This way, the current section presents a critical comparison between the traditional web services approach against the *Get*/*Set*-based patterns followed by several web standards for managing and monitoring resources. WS-Management specification [Arora et al., 2004] is given as example since it is considered a simple, lightweight and widespread standard for managing resources using web services.

When choosing the best approach to follow when deploying the proposed device model services it is essential to investigate the best approach on how to interact with these and also how to access device data. Two different approaches could be used for realizing such generic services: either use proprietary web services interfaces relying on common web services standards or employ an existing management standard, such as WS-Management.

WS-Management specification describes a general WS-* based protocol for managing systems such as PCs, servers, devices, web services, applications, and other manageable entities. To promote interoperability between management applications and managed resources, WS-Management identifies a core set of web service specifications and usage requirements that expose a common set of operations central to all systems management. This comprises the abilities to:

- Get, put (update), create and delete individual resources, such as setting parameters and dynamic data values.
- Enumerate the contents of containers and collections, such as large tables and logs.
- Subscribe to events emitted by managed resources.
- Invoke specific management methods with strongly typed input and output parameters.

In each of these areas of scope, the WS-Management specification defines minimal implementation requirements for compliant web service implementations. The implementation is free to be extended beyond this set of operations and the developer may also choose not to support one or more areas of functionality previously listed if that functionality is not appropriate to the target device or system. The WS-Management specification defines a standard form to access resources, but it does not define a resource description model. So, the user is free to define the XML resource description model that best fits its application, and share it so that others can interact with it.

Regarding traditional web services MEP, they should provide more complex services which will offer the user a more sophisticated level of information than simply get and set

data values. This approach must be avoided since it consists of replicating the functionality already available through management based solutions, such as WS-Management. The services should provide more intelligent and complex responses to the client. For example, instead of simply getting a collection of maintenance parameters, a maintenance service can provide a summary of activity focusing the most common points of interest, still based on those parameters but executing some processing over it before sending it to the invoker.

5.4 Service-oriented standards for Industrial Automation

5.4.1 Overview

As demonstrated in the state-of-the-art chapter, web services is currently the most promising approach concerning the application of SOA at industrial device level where the usage of high level service-based communications infrastructure promises to deliver completely innovative advances as firstly introduced by [Jammes and Smit, 2005b]. Web services is already a key technology to deliver software solutions that implement SOA across the different ICT layers of an enterprise. Such architectures would allow the use of web services both for peer-to-peer interactions at device level and for cross-layer integration with ERP, MES or SCADA applications.

Within this domain, two main specifications have emerged: DPWS [OASIS, 2009a], and OPC UA [OPC Foundation, 2008], a web services-based version of the OPC protocols widely deployed across industrial platforms. There is therefore a strong interest to provide a comparative analysis over these two currently concurrent technologies, identify their strengths and weaknesses alongside potential synergies and merging approaches.

After being introduced in Chapter 3 – section 3.2.6 as the two major specifications concerning the application of SOA related concepts and technology to the industrial automation domain, the goal here it to provide a comparative analysis over these in the domain of the application of SOA at device level, identifying possible synergies and merging approaches.

5.4.2 DPWS vs. OPC UA Assessment

It is essential to compare the main features of both OPC UA and DPWS specifications, in order to identify their strengths and weaknesses, but also to provide an overall assessment over their fitness to provide a definitive approach at device level in the service-oriented industrial domain.

OPC UA comes originally from low level device information field, it is still based on traditional object-oriented OPC information model and only partly employs web services protocols as a mean of communication. Due to this, OPC UA is sometimes accused of not being entirely SOA compliant. Also, OPC UA only provides limited extensibility, since it only allows methods to be added to objects and invoked through a predefined web

service. This does not allow a complete modelling of device functionalities as a network of services.

It is still important to refer other major advantages such as the easiness of communication through firewalls by using the web services protocol, improved security and efficient communication mechanisms, detachment from Windows OS, and, of course, the unified information model. Although OPC UA compliant products are still not widespread, based on previous OPC specification success it is expected to become some time soon a widely deployed standard.

Contrary to OPC UA, DPWS comes originally from high level ICT focusing at device level. Although based on a Microsoft specification, it is now a OASIS specification supported by open web standards. It is commonly recognised as the key specification to deploy SOA at device level [Jammes et al., 2005a]. DPWS does not specify any a priori information model to ensure openness, but at the same time it will require the implementation of methods to deal with content semantics.

A short summary of both specifications features is depicted in Table 5.5. Is it then clear that both specifications rely on very similar standard transport and messaging protocols. Besides those, and due to the performance constraints specific to this domain of application, OPC UA also specifies an additional set of optimised protocols not compliant with widespread SOAP/XML solutions. DPWS does not specify such optimised protocols, although the open and extensible nature of the SOAP 1.2 and WS-* specifications will allow the integration of such protocols in the stack without major disruptions.

OPC UA mainly relies on a client-server pattern, where the server is used to expose device-level information to higher level clients. Server and client stacks are usually not collocated, except in cases where servers are chained in a layered architecture, the server in one layer being the client of servers in the lower layer. DPWS, in contrast, relies mainly on a peer-to-peer pattern at the device level, even if the client-server pattern can also be used to provide access from high-level management layers. DPWS does not provide explicit support for integration with higher layers, although nothing prevents such a support to be added. WS-Management is a specification that addresses this topic and it can be integrated with DPWS.

DPWS is designed to work at the lowest device level and above, while OPC UA is more focused at gateway level which relies on proprietary communication protocols to access the low-level devices. Even if some UA profiles and stacks are already being embedded in low level devices, OPC UA remains focused on communication with SCADA or HMI applications – there is no real motivation to support peer-to-peer communication.

OPC UA specifies a rich and detailed meta-model, used by servers to model real-world objects and manage their relations, classes, attributes and variables. WS-Management does not specify a precise meta-model for the managed objects: it only addresses the external XML representation of resources. Having a well-defined meta-model is a clear advantage for developing OPC UA clients, as the knowledge of the exposed information models is clearly defined. On the other hand, the open approach of

Feature	OPC UA	DPWS + WS-*
Infrastructure		
General-purpose transport	HTTP 1.1	HTTP 1.1
General-purpose messaging	SOAP 1.2 WS-Addressing	SOAP 1.2 WS-Addressing
General-purpose encoding	XML	XML
Security	WS-Security WS-Trust WS-SecureConversation	WS-Security WS-Trust WS-SecureConversation
Optimized Transport	UA Native UA SecureConverstation	None (Open)
Discovery	WS-Inspection WS-Discovery UDDI	WS-Discovery
Architecture		
Software Architecture	Client-server Layered client-server	Peer-to-peer Client-server
Targeted hardware platform	Gateways/SCADA/HMI	Devices
Modelling		
Meta model	UA Object Model	None (Open)
Management		
Session Management	SecureChannel service set Session service set	None required (WS-SecureConversation can be used)
Resource Discovery and Selection	View service set Query service set	WS-Enumeration
Resource Access and Management	NodeManagement service set Attribute service set	WS-Enumeration
Eventing	MonitoredItem service set Subscription service set	WS-Eventing
Operation Invocation	Method service set	Standard Web Services

Table 5.5: OPC UA / DPWS features comparison

WS-Management supports greater extensibility and complete implementation freedom.

The specified management services are by nature different, as they represent the main purpose of each specification:

- By default, OPC UA uses stateful, connected, synchronous message exchange patterns. Lack of support for asynchronous and stateless communications limits the scalability and flexibility of the architecture in the case of peer-to-peer device interaction. Connectionless mode is generally considered more flexible and scalable, while stateless is recognised as one major cornerstone of the SOA paradigm.

- Since OPC UA defines a meta-model, the format of web services used to access and manage the objects implemented in the servers is rigid, with a limited number of extension points.

- OPC UA natively supports the semantics of historical data, both in the meta-model and in the management services, while this will have to be redefined by each device using WS-Management.

- Concerning event management, OPC UA provides a way to define monitored items

and subscribe to them. This way, clients subscribe to an event server when they are ready to receive events, and will then receive a new event when it is available at the server. Both DPWS and WS-Management employ WS-Eventing to provide a similar eventing feature defining by default a push strategy for notification delivery. This last strategy applies a generic eventing framework that can be deployed across different ICT enterprise layers.

It is widely recognized that reliability is crucial for the industrial market segment where the assumption that an event notification can be lost or not handled might be unacceptable in most of the cases. The OPC UA style of publish/acknowledge message pattern is most effective once the requirement for reliability is added into an eventing system not based on a message queue. However, improvements on network performance and employment of reliable message delivery protocols can dissipate this issue and ensure real eventing reliability using DPWS and WS-Management.

In summary, while OPC UA is only using WS–* protocols to address a traditional data model, DPWS allows the implementation of a true SOA approach promoting a device profile and hosted services.

5.4.3 Convergence

The previous assessment focus, on the one hand, over the strong similarities of the basic building blocks used by the two sets of specifications, and, on the other hand, the significant contrasts in their purpose and targets.

Due to these divergences, it becomes clear that any of these specifications does not entirely fulfill the requirements of the application of SOA at device-level in industrial domain. It is then imperative to refer the potential benefits that can emerge from a future combination of these specifications. These benefits can push new proposals and developments over this subject following a best of each world approach.

5.4.3.1 Potential benefits

Although the combination of DPWS and WS-Management can be a good candidate for a generic SOA framework for device level, such a combination will still have some limitations with respect to the functionalities provided by OPC UA for supervision and management. WS-Management does not define any specific meta-model for modeling devices, services or additional information to be managed. Any implementation of WS-Management for plant-floor devices should therefore specify an appropriate meta-model for resource modeling. OPC UA, on the other hand, provides such a meta-model well adapted to the modeling of plant-floor devices. More, upward compatibility between original OPC and OPC UA clients and servers is a strong argument for the quick dissemination of the OPC UA technology in the industry. WS-Management, on the other hand,

is coming from the IT and network management domain, and does not have the same image in the industrial world. For the above reasons, it appears that pushing OPC UA into the framework for device-level SOA will likely increase its adoption in the industrial domain [Hannelius et al., 2008].

WS-Discovery can also provide an important contribution to this framework by offering a mechanism to support standard peer-to-peer device discovery. WS-Discovery can be the bootstrap which will allow OPC UA devices to act as clients for other network devices.

Due to the strong similarities in the basic building blocks of both specifications, a joint implementation would bring several benefits:

- Avoid the duplication of effort and memory footprint in implementing a SOAP 1.2 stack, as well as other additional protocols such as WS-Addressing, WS-Discovery, WS-Security, WS-Trust and WS-SecureConversation.
- Since DPWS specification only comprises generic open web standards (not particular to any domain of application), it would ease the integration process across the diverse enterprise ICT layers. Although there might be a need to use specific OPC UA modules to some resource or performance constrained applications, an effort must be done to favour generic open standards as those defined by DPWS.
- An additional level of integration could be considered, by implementing the OPC UA meta-model in a way that will allow it to be accessed and managed through the WS of both OPC UA and WS-Management. This should be possible as WS-Management is very flexible about the resource model to be managed through its services.
- Exposing the same device meta-model to both OPC UA clients and WS-Management clients could ease the integration of the plant floor layer with the supervision layer, where OPC clients will be predominant, and the enterprise high level ICT, where WS-Management clients might be more widespread.

5.4.3.2 Merging approach

Driven by the envisaged benefits coming from a combination of both specifications joint with other complementary WS–* standards, a first convergence effort was initiated under the scope of SOCRADES project [SOCRADES, 2009]. A brief summary of the proposed solution is depicted in Fig. 5.6. This combined solution promotes mutual interoperability and paves the way to a device level interoperability compliant with SOA principles and technology.

In general terms, this approach integrates the best of DPWS and OPC UA in a common implementation starting with a DPWS specification and extend it with support for

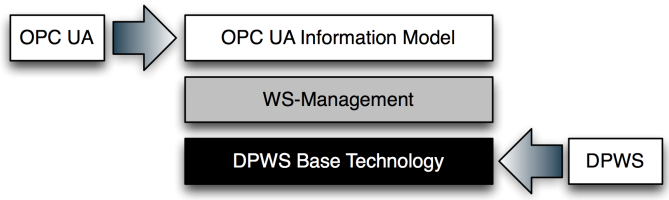


Figure 5.6: OPC UA / DPWS convergence approach overview

additional protocols required by OPC UA to provide security, but also UA Binary encoding and UA Native transport. The original OPC UA web services can be deployed as a set of common DPWS hosted services. The implementation of these services will follow the OPC UA meta-model specification. Since most of OPC UA embedded clients built today employ UA TCP, this protocol should be available as an add-on module in order to ensure backward compliance. It must be still noted that the majority of multi-domain SOA applications does not support this specification.

In the case of OPC UA, the meta-model is defined as an integral part of the specification, but its implementation remains an important issue. An OPC UA object model is made up of nodes, attributes and references, which are used to describe manageable resources. It is then necessary to define two-way links between the object model and the real device data and structures, and, finally, implement the concrete OPC UA web services that will allow the client to access and modify the model.

As WS-Management is designed to manage any external XML representations of resources, a simple mapping of the OPC UA object model onto a set of resources and their external representations would allow the WS-Management web services operations to manage the OPC UA object model through access and update of their external representation. WS-Management specification meets then some of the requirements to support the deployment and lifecycle management of devices and services, i.e. resources, by the end-user. In this context, the resources to be managed are logical devices (including physical device abstraction) and services (including *ServiceClass* instances). This way, the *Create* operation also defined by WS-Management will provision the install phase of the resource, the *Delete* operation, the uninstall phase of the resource. Furthermore, the specification is open to extend its core set of operations (*Create*, *Delete*, *Get*, and *Put* with custom operations like *Start*, *Stop*, *GetState*, etc. Using WS-Management standard, it would be possible to open traditional property-closed devices to external standard-compliant management tools. By having the ability to easily deploy these devices and its services into a physical device available on the network, the agility of the system is increased. This approach is generic enough to allow the integrator to implement its services with the programming language that best fits its current needs, define the service interfaces and then use it in a standardized manner through a service-oriented middleware as described in section 5.2.3.1. These generics services allow devices to evolve and

adapt their features and skills through the standard installation of new components in a more agile, open and transparent manner. The application will then be discoverable and interoperable in the network. The integrator has also the ability to manage the complete lifecycle of these resources in accordance with the evolution of production goals.

In order to cope with the constrained footprint requirements of some low level devices, it is fundamental that the integration of both protocol extensions and OPC UA web services remain completely modular. In particular, the use of the security protocols should not always be required, as the memory and performance requirements of these tasks are sometimes excessively exhaustive.

Another important input from DPWS specification is the introduction of WS-Discovery combined with WS-Addressing as mean to discover devices in peer-to-peer manner with no a priori knowledge of each device IP address.

5.5 Wrap-up

The proposed device model focuses on the specification of categories of resources that combined shape a service-oriented device model able to support above reference architecture requirements. This device model encompasses several built-in services that are expected to augment device added-value in terms of network openness, discoverability, interoperability, reconfiguration agility, along with simple and transparent resource lifecycle management. Concerning the management of resources, particularly devices and services, a complete state model and guidelines were presented. These management operations are available through the *Deployment* service, which can be considered mandatory in a dynamic service-oriented environment, i.e. it is embedded in the automation device as a built-in service. Other built-in services were also introduced to cover the scopes of setup, diagnosis, monitoring, as well as leaving the door open for proprietary services to cope with specific domains needs and distinguish their offer from its main competitors.

With this model in mind and taking into account the current state-of-the-art on service-oriented technological approaches, an implementation guidebook is described. A deep discussion about the two most relevant specifications related to the deployment of SOA concepts and vision into the device level in industrial automation is presented as well as a convergence solution between OPC UA and DPWS.

6

Implementation & Validation

The current chapter presents the work of implementation done within the scope of the validation of this thesis. The chosen validation process is discussed and applied taking into account the domain specifics and difficulties. Experimental setups are detailed by emphasizing the aspects of the reference architecture and device model being experimented. Finally, the relevant scientific contributions and peer validation are summarized.

6.1 Validation Approach

As discussed by [Creswell, 2009], research in general predominantly follows either qualitative or quantitative approaches to validate results although the number of mixed approaches is increasing and should be taken into account.

Quantitative approaches, normally perceived as the traditional scientific method, are intrinsically deductive and "*involve complex experiments with many variables and treatments*", alongside the elaboration of "*structural equation models that incorporate causal paths and the identification of the collective strength of multiple variables*". The two most employed strategies of inquiry for these approaches are experiments and surveys. Experiments include "*the random assignment of subjects to treatment conditions*" to extracted variables possible to be measured so that numbered data can be statistically analysed. Surveys "*include cross-sectional and longitudinal studies using questionnaires or structured interviews for data collection, with the intent of generalizing from a sample to a population*" [Babbie, 1990].

As counterpart, qualitative research approaches are mostly inductive where the nature of the data being processed is "*inductively building from particulars to general themes*" as discussed by [Creswell et al., 2003]. These approach are particularly aimed to domains that denote the following features: " *a) the concept is immature due to conspicuous lack of theory and previous research; b) a notion that the available theory may be inaccurate, inappropriate,*

Quantitative	Qualitative	Mixed Methods
+ Predetermined + Instrument based questions + Performance data, attitude data, observational data, and census data + Statistical analysis	+ Emerging methods + Open-ended questions + Interview data, observation data, document data. + Text and Image analysis	+ Both predetermined and emerging methods + Both open- and closed-ended questions + Multiple forms of data drawing on all possibilities Statistical and text analysis

Table 6.1: Quantitative, Qualitative and Mixed Research approaches procedures summary (compiled from [Creswell, 2009])

incorrect of biased; c) a need exists to explore and describe the phenomena and to develop theory; or d) the nature of the phenomena may not be suitable to quantitative measures". In the presence of a domain that includes one or more of these features, several research strategies can be employed, such as ethnographies, grounder theory, case studies, phenomenological or narrative research. Taking into account the nature of the current work, the case studies approaches emerges as the most suitable solution. During a case study approach, "*the researcher explores in depth a program, an event, an activity, a process or one or more individuals*" and "*collects detailed information using a variety of data collection procedures*" as discussed in [Stake, 1995].

Most research textbooks and journal articles balance between qualitative and quantitative research methodology despite the existence of substantial literature to support the use of mixed methods as discussed by [Mackenzie and Knipe, 2006]. A wider acceptance and employment of the mixed method research can only enrich and strengthen research approaches through the application of qualitative and quantitative methods in complementary ways. Mixed method, like all research approaches, needs to be considered through a critical lens to check its adequacy to the domain of application and legitimacy as a research contribution. The paradigm framework should define the choice of methodology, although sometimes this factor is not often addressed effectively as also referred by [Mackenzie and Knipe, 2006]. A summary of the research approach procedures is depicted in Table 6.1

The nature of the proposed research suggests a mostly qualitative validation approach, which is naturally reflected in the implementation done in the scope of the present work. Furthermore, the validation itself in this context is a complex problem hardened by the multidisciplinary nature of this work.

As defined by [Lankhorst, 2009], the "*architecture is the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principle guiding its design and evolution*". As backed by [Schekkerman, 2003] and since SOA has its origin in the business ICT domain, the design of an enterprise architecture entails a conflict between business and ICT workforces. There are evident benefits for

organizations that are able to put together the available range of options into a holistic enterprise architecture framework of flexible domains and supportive technology that effectively align business and ICT resources and processes.

The field of software engineering includes research disciplines that focus on preventing and remedying malfunctions and ensure predefined behaviour. In this area the prevalent validation approach consists of software testing. However, as stated by [Bertolino, 2007], this approach is still largely ad hoc, expensive, and unpredictably effective. Nevertheless, software testing will continue to be an essential activity for software engineering. Despite the continuous advances in the way it is built and employed, the software will always need to be eventually verified and monitored. When there is a need to test and validate a technology system or component, the black box and white box testing as introduced by [White, 1987] are still some of the most common solutions. Black box testing, also referred as "input-process-output", focus on the system functionality in terms of expected versus verified results taking into account the same set of input parameters. White box testing is more suitable for troubleshooting phase since it encompasses the system structural view and allows the verification of the response taking into account the participation of the related system components. As also remarked by [Bertolino, 2007], by only focusing on the specific problems of software, there is a risk to overlook some central aspects of the system as a whole.

As stated by [Skadron et al., 2003], it is becoming harder and more time-consuming to construct accurate models of modern computer-based systems. All models are abstractions of the original source that are based on many different types of approximation as discussed by [Oberkampf and Roy, 2010]. As the range of requirements, approaches, technologies and domains of applications becomes more diverse, creating a suitable set of available benchmarks and metrics becomes a more challenging or even unbearable task. Further, the substantial effort required to develop highly reliable simulation tools are commonly not followed by the according academic interest due to the lack of expected rewards.

In the scope of simulation, validation is "*the process of determining the degree to which a computational model is an accurate representation of the real world*", as in [Oberkampf and Roy, 2010]. Researchers struggle while pursuing advanced investigations when they are limited to a framework where no benchmark tools or recognized metrics are available. There are, to the author's knowledge, no concrete benchmark studies on SOA-based approaches related on their fitness to a particular domain, specially to the industrial automation device level. There is no quantitative approaches available on how SOA software tools and methodologies are suitable for this domain, neither a procedure to validate its significance and results. Even if several tools exist to model, simulate and check an architecture for its consistency in terms of reliability and feasibility there is no way to determine if it really fits domain requirements and compare with concurrent approaches besides employing qualitative considerations.

In the scope of the present work white box and black box testing solutions are only

suitable for validating the implementation of each element of the architecture and check for its feasibility and behavioral results – black box testing for atomic components and white box testing when dealing with components collaboration while performing a particular activity.

In the scope of the current work, the validation approach focused on developing prototype implementations that covered the key aspects of the proposed reference architecture and device model in terms of functionality and interaction with the remaining entities. Particular software optimizations or extensive testing was not pursued due to the academic nature of the thesis work and its test-bench validation purposes.

Most applications developed today, and particularly those based on SOA technology, rely on a predefined middleware solution which supports the interaction between components or enables the access to resources. Choosing the right platform for the current application is always an essential question to be posed. As observed by [Baresi et al., 2003], it is common to start by developing a prototype implementation covering all key application scenarios and operations, like components communication, user interaction, access to remote components, behavioral flow, etc. Therefore the model chosen to specify the system architecture needs to be understood and validated by domain experts with little or no background in formal specification.

As also depicted by [Baresi et al., 2003], an explicit visual representation of the architecture in some diagrammatic language like the UML [Rumbaugh et al., 2004] is often regarded as helpful, even if there is a risk to trade this intuitive nature for ambiguity. In this latest study, the authors present an approach for modeling and analyzing software architectures based on modeling the architectural style through class diagrams and the dynamic behavior using a graph transformation system. An architecture compliant with the style can be regarded as an instantiation of the class model. The whole approach was applied to an SOA use case scenario to better exemplify the presented concepts, however besides focusing on the modeling using extended UML class diagrams the dynamic part of the system was verified by employing graph transformation as a visual, yet formal approach to model and reason about these architectures. However, once again, and although diagrammatic language reveal to be valuable when modeling the system based on a reference architecture, in the context of the present work the effort is put on its fitness to the domain of lifecycle device support in industrial automation and verification of the gains in terms of adequacy, feasibility and improvement of systems integrator quotidian routines.

Chapter 4 introduces the architecture and its elements alongside a collection of use cases that demonstrate its contributions to the domain. The prototype implementations will hold to these to demonstrate the feasibility and adequacy of the proposed architecture. In relation to the device model presented in Chapter 5, it was also prototyped and deployed into an authentic industrial automation installation in the scope of an international project demonstrator. Both experimental setups are detailed in terms of development decisions and outcomes, identifying at the same time the aspects of the architecture

Dimension	Instrument	Shows that	Chapters (Ch) / Sections (Sc)
Feasibility	Physical Prototype	Both the reference architecture and device model can be implemented and deployed	Ch. 4 in Sc. 6.2.2, Ch. 5 in Sc. 6.2.1
Relevance	Furtherance of domain of interest	The existence of research gaps and potentials as shown in the presented survey and discussion about the application of SOA principles and technology to contemporaneous industrial automation device level	Ch. 3
	Research Follow-up	Several initiatives are interested to pursue the research of SOA related aspects and applications to several industrial areas,	Ch. 6 - Sc. 6.3.2
	Use of contemporaneous standards	Current major domain standards as DPWS and OPC UA can still be improved by combining their key aspects into a convergent solution	Ch. 5 - Sc. 5.4.2
Adequateness	Physical Prototype	The proposed approach can be deployed and executed on a real industrial automation installation	Ch. 5 - Sc. 6.2.1
Peer Acceptance	Integration with International R&D projects	The scientific and industrial technical aspects of this work are accepted and recognized	Ch. 1 - Sc. 1.5
	Publications in International peer reviewed Journals and Conferences	The work is accepted and recognised by peers from the domain and can be further investigated and developed	Ch. 6 - Sc. 6.4

Table 6.2: Summary of the validation dimensions and methods

and device model being validated in each step.

Another important aspect of the employed validation process relates to peer evaluation and acceptance both by academia and industrial partners. The acceptance of the proposed work by peers certifies that the research is relevant to the application domain and that it can be reused in future research and developments.

Table 6.2 summarizes the principal dimensions of the validation procedure, the methods used for that purpose as well as the associated sections on this thesis.

6.2 Experimental Setup

In the context of the current work, two distinct experimental setups were developed to cover the key aspects of this thesis contribution. The reason behind the existence of two separated experimental setups relates with logistic aspects and access to the industrial automation platform used during ITEA SODA project. The first phase of the present work was executed in the scope of the author's participation in this project as part of Schneider Electric *DInnov* team at 38TEC site in Grenoble, France where this industrial platform is located and also used as a test-bench for several other company prototypes and solutions. The last phase of the work took place at the Electrical and Computers Engineering

Department at FCT–UNL campus in Caparica, Portugal. In this period, the author, due to easiness of access and availability, employed an existing educational kit that simulates a flexible industrial production environment and adapted it into a service-oriented production scenario in order to test and validate the remaining architecture components. Nevertheless, this fact helped corroborate the legitimacy of the approach by applying it into an additional automation scenario.

As referred before, the first experimental setup was developed in the scope of SODA project industrial demonstrator and focused on the test and validation of:

- Application of the device model on low-power smart devices and legacy equipment.
- Employment of domain programming languages to specify the behaviour of services hosted by devices.
- Service design and deployment procedures using DPWS and WS-Management.
- Use of wireless tools for setup and monitoring
- Integration with MES/SCADA systems through service-oriented communication.

The second experimental setup was deployed into the *MOFA France* education kit and explored the aspects related with test and validation:

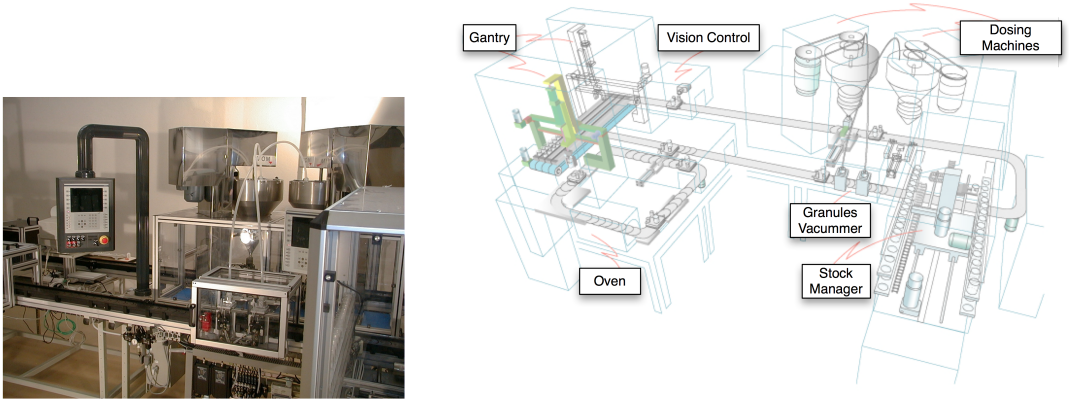
- Device explorer functionality, including the dynamic retrieval of logical device topology, semantic service matching and translation, as well as the deployment of semantic gateways.
- Device and Services ontology and Semantic Assistant.
- Management and Monitoring of service-based process plans.

6.2.1 ITEA SODA Demonstrator

6.2.1.1 Platform Overview

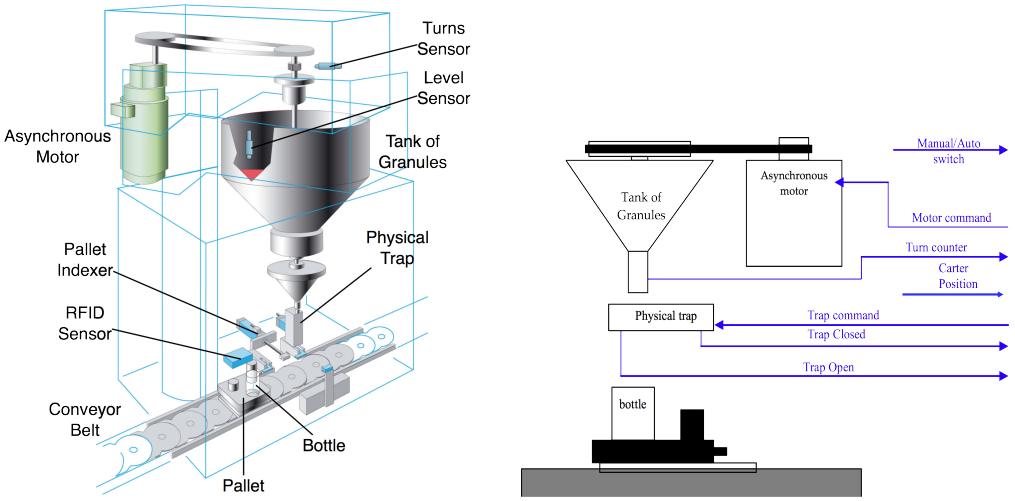
This industrial automation installation represents a demo packaging production system that releases orange and white granules into small recipients attached to pallets that run over a circular conveyor (see Fig. 6.1). The original control system consist of a central PLC that controls the overall system by coordinating a group of several distributed I/O modules connected using *CANopen* technology.

In the scope of SODA Project, the main focus was put on the dosing area where two dosing machines that are connected to the transport line fill the recipients of white or orange granules, respectively in accordance to current pallet Radio-Frequency Identification (RFID) tag information (see Fig. 6.2).



(a) Platform photo (b) Platform elements detail

Figure 6.1: SODA project industrial automation platform



(a) Dosing Machine detail (b) Dosing Machine I/O

Figure 6.2: SODA platform – Dosing machine details

In order to support the deployment of a service-oriented solution several modifications were made to the original system. The existing PLC code needed to be extended to support a *SODA Mode*. When active, this new mode of execution transferred the control of the dosing machines and their components to the prototype versions of *FTB* I/O devices, which implemented a big part of the device model proposed in Chapter 5. Logical devices and services were deployed into this set of *FTB* devices to control both the motor and trap in a coordinated way to build-up a *DoseMaker* device. The PLC remains responsible for the control of the pallet indexer, conveyor beld, *Ositrack* RFID system, as well as all the other system components that were not considered for the actual test-bench. The deployed services related to the dosing area interacted using DPWS software stack. Whenever a pallet tag is detected by the RFID system at the dosing area entry (*PalletID WS*), it will invoke an indexer service (*Indexer WS*) to take the pallet from the conveyor

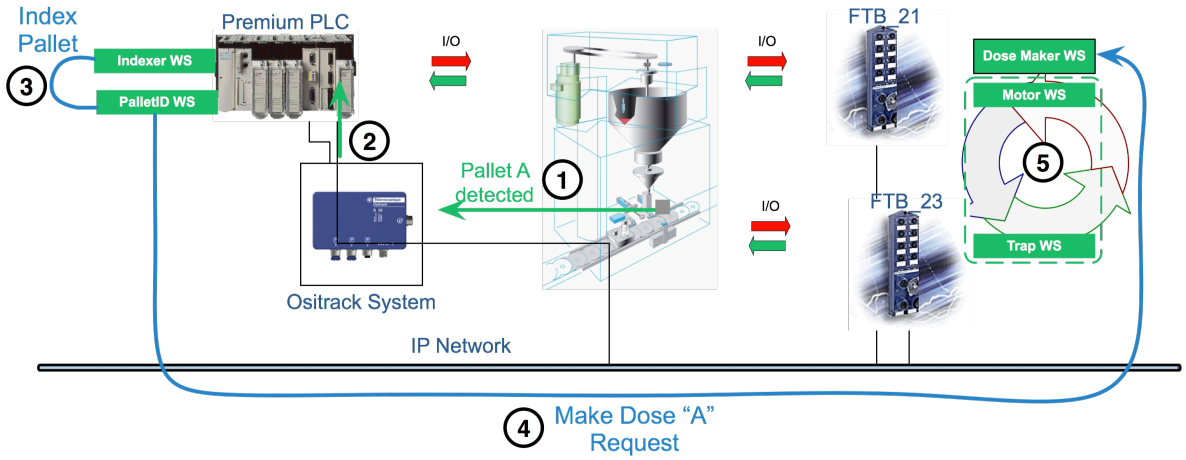


Figure 6.3: SODA experimental setup – Example of services interaction

belt and position it ready to be filled in the according dosing area. Then, the dosing service (*Dose Maker WS*) service is invoked and the recipient is filled with granules of the according colour and quantity. Subsequently, the indexer service receives the end of execution notification from the dosing service, the pallet is again placed in the line and the process restarts with the next pallet in line. Fig 6.3 exemplifies this service interaction pattern.

In a broader scope, the SODA industrial demonstrator involved the participation of other partners besides *Schneider Electric* that provided the industrial platform itself and the automation equipment. *Geensys* provided a modified version of its *Control Build* [Geensys, 2009] software automation platform to support the creation of SOA system based on SCA design and low level implementation using IEC61131 languages. Also, an extended version of the *ARC Informatique PcVue* [ARC, 2009] HMI/ SCADA software was used to access monitoring data directly from the devices involved in the dosing process in a service-oriented manner and present these to the user in a graphical interface that mimicked the current process state. Finally, *CapGemini* integrated a smartphone application that included a light version of a device explorer to allow the mobile discovery of devices in the network and connect to them to retrieve their current status and metadata, as well as to test of its hosted services via wireless. The complete SODA industrial demonstrator is depicted in Fig. 6.4.

The involvement of these project partners clearly demonstrates the interest of companies from different sectors to come together and pursue a common service-oriented approach that unifies the interaction between the several dissimilar network entities in a transparent and standard way.

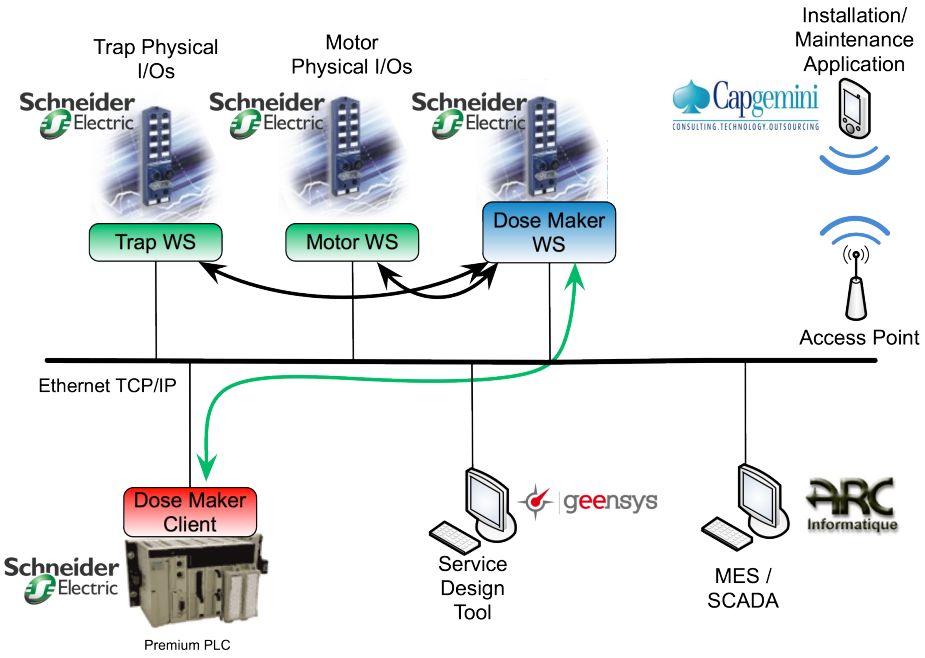


Figure 6.4: SODA Demonstrator overview

6.2.1.1.1 Adopted Technology

The implementation of this experimental setup involved the use of several technology packages and equipment. Although supplementary details on each experimental setup segment will be presented at the according chapter, all communications relayed on the open source *SOA4D* DPWS stack [SOA4D, 2009a] and WS-Management protocol implementation [SOA4D, 2009b]. The device firmware included the implementation of DPWS and WS-Management specifications in a joint solution to support the deployment of services, besides other features already specified in Chapter 5. DPWS extends the set of core protocols associated with the application of web services (IP, TCP, UDP, HTTP, SOAP), with others such as WS-Discovery, WS-Eventing, WS-Addressing, WS-Security, WS-MetadataExchange, et. al. Also, as presented in section 2.2.3.1, SOAP-based implementations are currently the most employed when addressing industrial automation device level. Nevertheless, the current abstract architecture and device model is not constrained to SOAP or any other particular specification and can be implemented with the technology that best fits the developer and the application in study.

Using DPWS, scalability is favoured by the fact that event-driven communications are considered substantially more efficient, in terms of bandwidth usage and processing constrains, than polling-based communications. Furthermore, by exploiting WS-Discovery the process of discovering devices in the network in a distributed peer-to-peer manner significantly increases the agility when dealing with a mutable device setting.

After Schneider Electric and other European projects partners such as ABB, SAP and Siemens strong push to put DPWS-based middleware at device level, other automation

companies such as Beckhoff [Beckhoff, 2008] are already adopting the specification both for industrial and building automation scopes. It is also important to refer that this solution was possible of being implemented in modified versions of Schneider Electric *FTB* devices which are considered low power devices due to their very modest hardware specifications: ARM STR9 processor with 512 KB of Flash memory and 96 KB of RAM. It as also used a Schneider Electric PLC Modicon TSX Premium in addition to a TSX ETY5103 Ethernet network module running a VxWorks OS, which was modified to hold a DPWS client.

6.2.1.2 Device Model

As initially discussed in sections 5.3 and 5.4, WS-Management allied with a DPWS stack revealed to be a good technology option concerning the implementation of the proposed service-oriented device model.

In summary, the application of WS-Management at device level enhanced the agility and openness of the traditionally morose and complex task of managing a wide range of network devices in a heterogeneous service-oriented environment. The WS-Management specification turned out to be the fittest approach, in comparison with traditional Web Services, to manage devices due to its inherent nature and extensibility easiness. There is no need to replicate the same functionality already provided by WS-Management – proprietary web services approach should only be applied if more sophisticated and complex management features are needed. The methodology to deploy and manage resources embedded in a physical device lies over an open standard possible to be employed to different levels of the complete system architectures – unified management approach.

As the current implementation employs DPWS as service-oriented middleware, the device model is then used to describe DPWS devices and includes the necessary information to provision discovery and metadata exchange processes. It also contains subsections describing hosted services, which are instances of the service classes described above. The outline of a device model element is presented in Listing 6.1. The device model besides including a collection of information related with its features and hosted services, it also includes a dynamic part to support the management of new resources deployed into the device. By exploiting the built-in deployment service is possible manage and control the lifecycle of devices and services, i.e. install, uninstall, start and stop.

Listing 6.1: Device element outline

```
<dd:Device ...>
  <dd:Address>xs:anyURI</dd:Address>?
  <dd:Types>list of xs:QName</dd:Types>?
  <dd:Scopes>list of xs:anyURI</dd:Scopes>?
  <dp:ThisModel ...>
    <dp:Manufacturer ...>xs:string</dp:Manufacturer>+
    <dp:ManufacturerUrl>xs:anyURI</dp:ManufacturerUrl>?
```

```
    <dp:ModelName ...>xs:string</dp:ModelName>+
    <dp:ModelNumber>xs:string</dp:ModelNumber>?
    <dp:ModelUrl>xs:anyURI</dp:ModelUrl>?
    <dp:PresentationUrl>xs:anyURI</dp:PresentationUrl>?
    ...
  </dp:ThisModel>
  <dp:ThisDevice ...>
    <dp:FriendlyName ...>xs:string</dp:FriendlyName>+
    <dp:FirmwareVersion>xs:string</dp:FirmwareVersion>?
    <dp:SerialNumber>xs:string</dp:SerialNumber>?
    ...
  </dp:ThisDevice>
  <dd:Service serviceId="xs:anyUR"?>*
    <dd:ServiceClass classId="xs:anyURI"/>
    <dd:ServicePort>*
      <wsa:Address>xs:anyURI</wsa:Address>
      <wsa:ReferenceParameters>...</wsa:ReferenceParameters>?
      <wsa:Metadata>...</wsa:Metadata>?
    </dd:ServicePort>
    <dd:Reference name="xs:NCName">*
    [
      <wsa:EndpointReference>
        <wsa:Address>xs:anyURI</wsa:Address>
        <wsa:ReferenceParameters>...</wsa:ReferenceParameters>?
      </wsa:EndpointReference>
      |
      <dd:DiscoveryHints onMultipleMatches="pickOne|fail"
                         bindingTime="deployment|runtime"
                         onReferenceLost="retry|ignore|fail"
                         serviceId="xs:anyURI"?>
        <dd:Hint>+
          <dd:Types>list of xs:QName</dd:Types>?
          <dd:Scopes>list of xs:anyURI</dd:Scopes>?
        </dd:Hint>
      </dd:DiscoveryHints>
    ]
    </dd:Reference>
    <dd:PropertyValue name="xs:NCName">*
      ...
    </dd:PropertyValue>
  </dd:Service>
</dd:Device>
```

WS-Management specification revealed to fit the present need: the resources being devices and services, as defined in 5.2.3.1.1. For example, the *create* operation from WS-Management allows the install phase of the resource, the *delete* operation, the uninstall phase of the resource. Furthermore, the specification is open to extend this core set of operations (*create*, *delete*, *get*, and *put*) with custom operations like *start*, *stop* or *getStatus*, as presented in section 5.2.3. By exploiting WS-Management specification these devices

are now accessible by any external tools also compliant with this open web standard.

6.2.1.2.1 *ServiceClass* model

The *ServiceClass* model is used to describe service implementations. This solution was inspired by the SCA implementation model with some simplifications. A service implementation is characterized by the set of service *portTypes* it provides as described in common WSDL files, the set of references to services that it may require and configurable properties that regule its behaviour. The *ServiceClass* model contains an additional element (*Implementation*) used as a placeholder for technology-specific implementation data. This element can include a class name and optionally a *jar* file for Java implementations, an entry point and a Dynamic Link Library (DLL) on Windows or Linux, or a program and some initialisation data for interpreters. Each technology will define the actual element structure required to hold the implementation figures. The current implementation employed an interpreter-based solution to support the deployment of IEC61131-based code as depicted in a subsequent section. The outline of the *ServiceClass* element is presented in Listing 6.2.

Listing 6.2: *ServiceClass* element outline

```
<dd:ServiceClass classId="xs:anyURI" ...>
  <dd:Interface name="xs:NCName"? type="xs:QName"/>*
  <dd:Reference name="xs:NCName" type="xs:QName" mustSupply="xs:boolean"?/>*
  <dd:Property name="xs:NCName" type="xs:QName"
               mustSupply="xs:boolean"? multiple="xs:boolean"?>*
    default-property-value?
  </dd:Property>
  <dd:WSDLInfo targetNamespace="xs:anyURI" location="xs:anyURI"/>*
  <dd:Implementation/>
</dd:ServiceClass>
```

6.2.1.3 Service Design

In the industrial automation domain, systems integrators are used to their own processes and programming languages. IEC61131-3 programming languages are still the most used within this domain due to its simplicity, run-time performance and availability of compliant equipment. The challenge is then how to turn this reality compatible with an ample and unified SOA environment.

As referred before, *Geensys* provided a modified version of its *Control Build* software automation platform to support the development of SOA system based on SCA design and allows systems integrators to abstract different system components and coding their behaviour using IEC61131-3 programming languages. The *Geensys* tool also permitted the export of these components into PLCopen XML [PLCopen, 2009] format files that can be then used as service implementation when deploying a new service into a device as described in the following section.

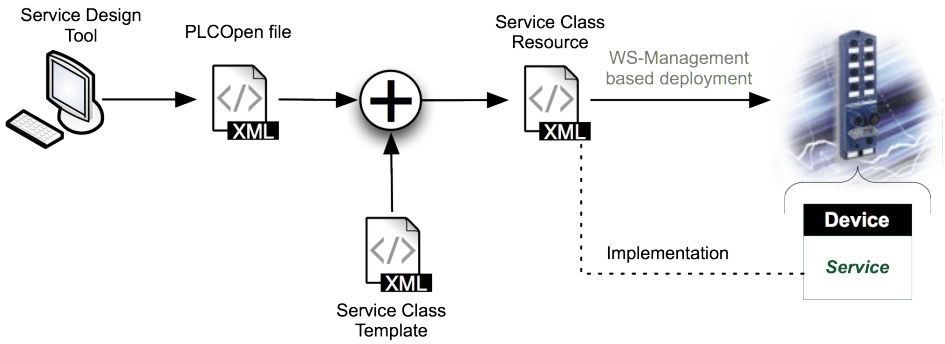


Figure 6.5: Dynamic deployment process

The application design and programming phase is entirely independent from the deployment phase where the automation engineer maps the different resources to the target automation device. It is possible to deploy the same application each time into a different range of devices, since the designed application maintains its abstraction and reusability. Since it is possible to deploy the application into several distributed devices from a single point, instead of physically connect to each individual device this approach enables a faster deployment phase. Even though this implementation focused over industrial automation domain, it remains abstract enough to adapt to other domains of application since the implementation details are isolated from the process to deploy new services into devices.

6.2.1.4 Services Deployment

The deployment service is itself a built-in service to allow the integrator to deploy its own resources i.e. logical devices and its services. Since only the *Service Class* resources comprise implementation details, the integrator can employ the preferred language regarding current application specificity and requirements, while devices and hosted services interfaces remain abstract and clearly separated from the hardware part.

As proof-of-concept experiment, it was possible to use IEC61131-3 languages to describe and control machine behaviour. These control behaviours were then abstracted as logical devices possible to be discovered as any other DPWS device – the invoker only cares about service functionality and not how the service is implemented. As referred before, the IEC61131-3 code was translated to PLCOpen XML format by the service design tool and added as an element of the according *ServiceClass* resource, as presented in Fig. 6.5. By embedding the original PLCopen code as an element of the *ServiceClass* instance, this resource is now possible of being deployed into a device supporting services deployment as specified in previous section 5.2.3.1 and act as the implementation of a service hosted by that device.

As referred before each implementation technology needs to redefine the *Implementation* element within the according *ServiceClass* element. In the case of the IEC61131

engine, two types of configuration data are possible: mapping between web service messages and IEC61131 variables, and an IEC61131 program and related configuration data. The outline of an implementation element encompassing IEC61131 code is depicted in Listing 6.3.

Listing 6.3: *Implementation.IEC61131* element outline

```
<iec:Implementation.IEC61131>
  <iec:Mapping refSeparator="iec:Character":'.'?
refIsPrefix="xs:boolean":true?>
    <iec:PortType name ="xs:QName" role="service|ref|both":service?>+
      <iec:Operation name="xs:NCName" variable="xs:NCName"
isEvent="xs:boolean":false? eventSources="list␣of␣xs:NCName"?>+
        <iec:Input tag="xs:QName" action="xs:anyURI">?
          <iec:Parameter name="xs:QName" type="xs:token" variable="xs:NCName"/>*
        </iec:Input>
        <iec:Output tag="xs:QName" action="xs:anyURI">?
          <iec:Parameter tag="xs:QName" type="xs:QName" variable="xs:NCName"/>*
        </iec:Output>
      </iec:Operation>
    </iec:PortType>
  </iec:Mapping>
  <iec:Program>xs:base64Binary</iec:Program>
</iec:Implementation.IEC61131>
```

Once the component is made available and active, it will be possible to retrieve it and its services invoked or events subscribed in the network as any other DPWS device. It should be noted that this template is independent from particular implementation details, which will be comprised in a parameter value that can be run or interpreted by device firmware.

All the information related to devices and according services available on a physical device are stated in a XML file – XML Device Configuration File that can be accessed through a WS-Management client application as depicted in Fig. 6.6. This file contains the elements that describe *ServiceClass*, *Devices* and *Services* instances deployed in that particular physical device. This file will be updated every time a new resource instance is deployed or modified. Although stored in the same file as deployed resources, the built-in services are protected from external modifications by adding a read-only tag to built-in resources.

This implementation was executed in the context of SODA project industrial demonstrator, being also consequently employed during SOCRADES project industrial pilot application.

6.2.1.5 Integration of Legacy Equipment

In order to allow a natural integration of the new dosing area approach within the existent industrial installation there was a need to execute a few updates to the existing

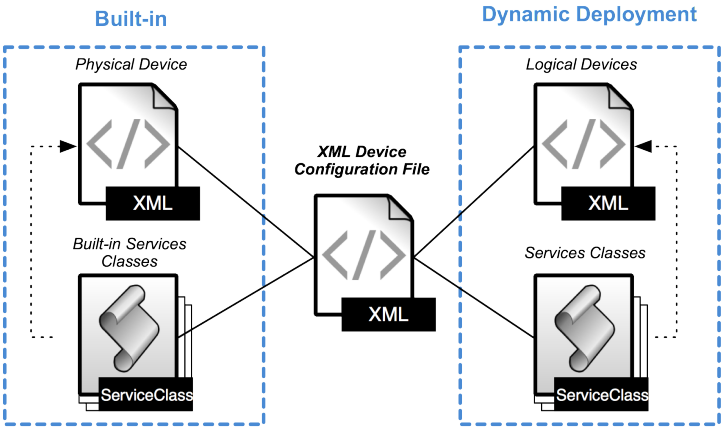


Figure 6.6: XML Device Configuration file

PLC system to integrate it in a new service-oriented application. These developments exemplify a possible solution on how to integrate legacy equipment in the scope of a new service-oriented approach using a software-wrapper.

Although the dosing area is controlled and abstracted by a group of *FTB* devices, the *Dose Maker* service needs to be invoked whenever a new pallet is in place to be filled with the according granules dose. Since the existing Modicon TSX Premium PLC controlled the conveyor belt, RFID system and pallet indexer there was a need to make this PLC capable of invoking the *Dose Maker* service whenever a new pallet is set to be filled. A TSX ETY5103 Ethernet server module was added to the PLC rack to support this new feature. The original module relays on a firmware implementation based on a VxWorks OS and allows the access to the PLC I/O and tables of variables. The original firmware was modified using *Tornado* 2.0.2 development suite and it was created a software abstraction layer to support the use of the DPWS C stack in the VxWorks OS and implement both a logical device that abstracts the PLC as a physical device in the network and implement a *DoseMaker* client. The new components files were downloaded into the modified *ETY* Ethernet server module and the bootstrap loader configuration file was modified to launch this new resource whenever the PLC is started as depicted in Fig. 6.7.

The implementation of the *DoseMaker* service client embraced the monitoring and control of some internal PLC variables by using a Modbus API to access these. As outlined in Fig. 6.8, these variables determine when a pallet is set to be filled to invoke the *DoseMaker* service, wait for the reply, put the pallet back to the conveyor belt and wait for another pallet. This component is only active whenever the SODA mode is on.

The current software-wrapper approach proved valid to cope with legacy equipment that does not originally comply with the requirements of a new service-oriented application. If a modification of firmware is not possible, a gateway-based solution should be employed.

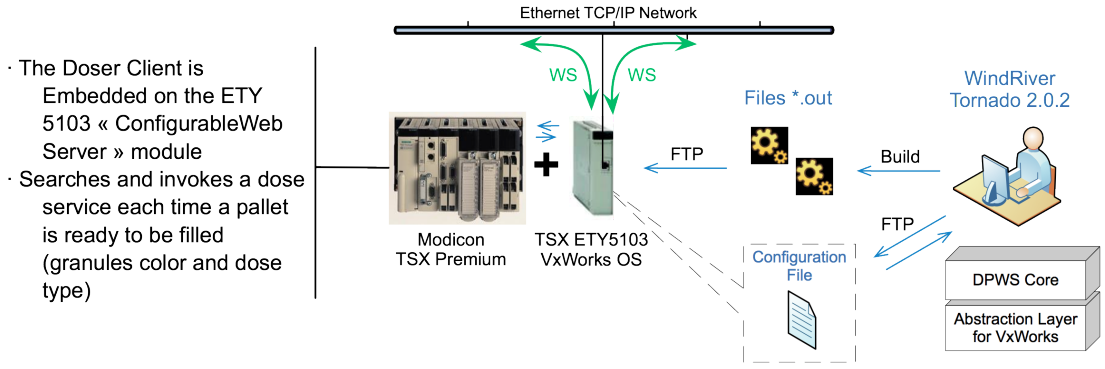


Figure 6.7: Software-wrapper solution for the PLC system

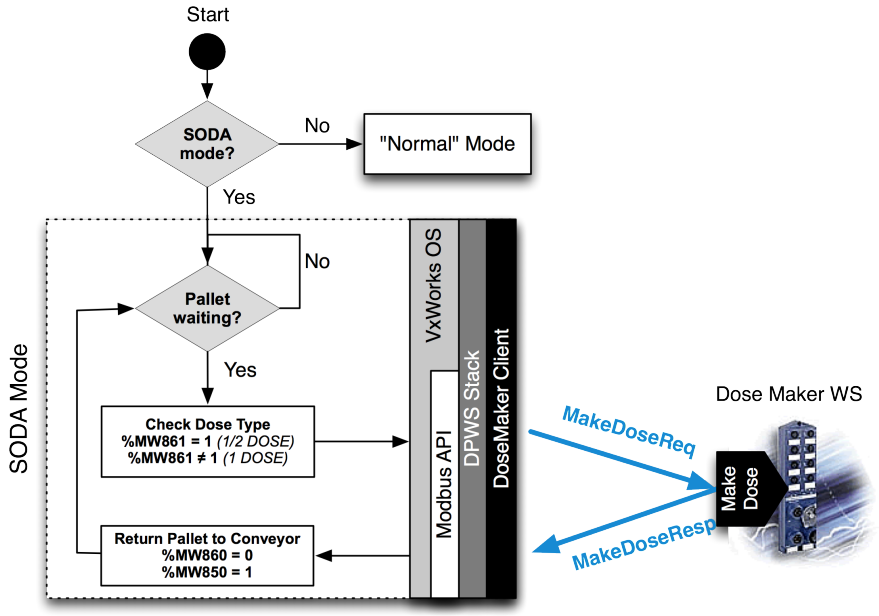


Figure 6.8: Ethernet server module implementation detail

6.2.1.6 Wireless Discovery and Setup

In the context of the current domain of application and comparing with traditional setup solutions that involve a physical connection to each singular device using proprietary cables, protocols and tools, the current solution allowed the movement of the systems integrator along the shop floor using a mobile device, such as a smart phone or tablet with a wireless connection while interacting with the existing devices visible in the network.

The prototype developed by CapGemini included an application running on a smartphone that implemented both DPWS and WS-Management to allow a peer-to-peer device discovery, access and testing. As case study example it was considered the process of installation and setup of a new device in an industrial automation environment, in this case, a *FTB* device. After cabling a new *FTB* I/O, setting its identification number using

the integrated "coding wheels", connecting it to the network and powering it on, this new *FTB* becomes available and possible of being discovered in the network. Afterwards, the system integrator uses its mobile application to perform a network search and detects the previous plugged device by its identification number. The number introduced in the *FTB* "coding wheels" will serve as the suffix of the device *FriendlyName* default value. Besides retrieving its metadata, the systems integrator can get the device status and test each I/O item by modifying the appropriate item value at the device resource model and see its effect on the real system locally.

6.2.1.7 SCADA Connection

In context of the dosing system, a SCADA HMI application was put in place to acquire information from the system and supervise the current service-oriented application in a graphical way. By exploiting WS-Management to have access to the *DoseMaker* resource, as well as its constituents *Motor* and *Trap* it is possible to retrieve some items through polling whenever needed and be notified when predetermined events occur. The SCADA system will subscribe to the event sources relevant to the current application and provide the handlers to cope with the received events.

As referred before, it was employed an updated version of the ARC *PcVue* software that embodied a WS-Management implementation and allowed the mapping of device resource model items to internal software variables. These variables controlled the behaviour of the user HMI accordingly to face the current dosing area state. Table 6.3 shows the items used by the SCADA system to mimic the current system state of the *DoseMaker* resource.

Item	Type	Access	Usage
Status	Enum	Read only	+event
NumberOfDoses	Integer	Read only	HMI Request
DoseAverageTime	Float	Read only	HMI Request
DoseMaxTime	Float	Read only	HMI Request
DoseMinTime	Float	Read only	HMI Request
CarterStatus	Enum	Read only	+event
TankLowLevel	Enum	Read only	+event
ProcessStatus	Enum	Read only	+event
Manufacturer/vendorName	String	Read only	Identification
ModelName/productFamily	String	Read only	Identification
ModelNumber	String	Read only	Identification
ProductName	String	Read only	Identification

Table 6.3: SCADA monitoring items for *DoseMaker* resource

The current solution revealed to be efficient and easily adaptive to the SCADA domain by supporting both polling- or event-based access to application resources via an open web standard specification. By discovering devices in a peer-to-peer form, retrieving their resource models and directly map these into the actual SCADA system the current approach was considered relevant as a new mean to acquire data from contemporary industrial automation systems and provide direct transparent access to each device.

6.2.2 MOFA France educational kit

6.2.2.1 Platform overview

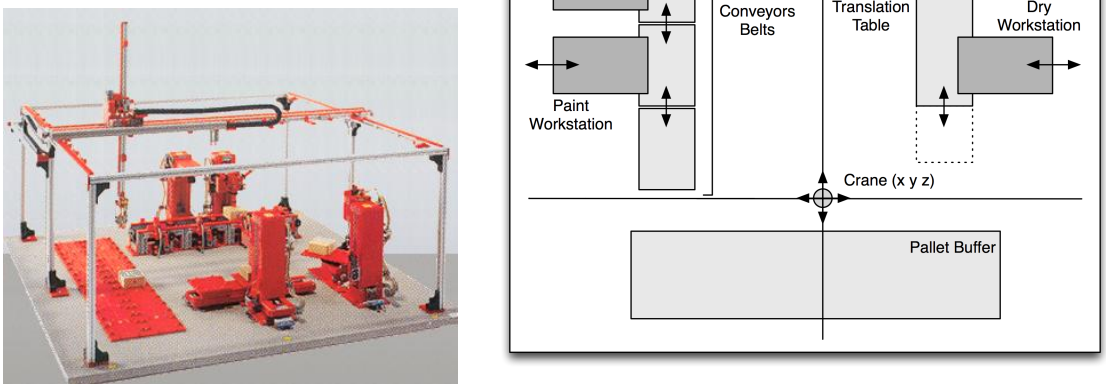
The MOFA France kit by Staudinger GmbH [Staudinger, 2012] simulates a flexible manufacturing system as a multi-path closed loop manufacturing circuit based on generic manufacturing tasks (see Fig. 6.9(a)). Staudinger models follow the modular Fischertechnik-based concept to replicate complex industrial projects in close-to-reality details. This allows systems integrators to easily discover potential problems during planning and programming, while testing suitable alternatives that can be verified quickly in a controlled environment. As verified by [Barata et al., 2008], this educational platform proved to be extremely useful for testing purposes since past experiences showed that a lot of effort can be saved if an educational platform is used. Still, key aspects of a real industrial environment should be taken into account, such as real-time and reliability, resources concurrency, mechanical and electrical relations, etc. Of particular relevance is the easier addition and removal of physical modules, which is much easier with this type of platform than with a real industrial system for obvious logistic reasons.

This educational kit is composed of four machines that may be adaptable to different types of manufacturing tasks, a buffer area, a crane robot, local transporters (conveyors or tables) and sensors to detect pallet positions. For this particular case study, the tasks that can be performed in these machines are *Weld*, *Paint*, *Dry* and *Drill* as presented in Fig. 6.9(b). The pallets are represented by wood blocks with a carved metal ring to activate the positioning sensors. They represent pallets with product parts, subassemblies, or even raw materials that need to be transformed or processed. These pallets can be stored in the buffer area or being transported by the crane to the available loading positions to be processed.

6.2.2.1.1 Equipment

The original control equipment, composed by a legacy data acquisition board installed in a PC that allowed the access to the kit I/O using a C++ API. was replaced by a distributed service-oriented PLC solution. This new control solution was composed by a distributed collection of Inico S1000 modules. The Inico S1000 is a smart Remote Terminal Unit (RTU) capable of real-time control, field data processing, web-based monitoring and integration with SCADA/HMI systems. The S1000 hardware configuration includes a 32-bit CPU running at 55 MHz and 8 MB of available flash memory, 10/100 Mbit Ethernet port, 8 digital inputs and 8 digital outputs.

Besides handling typical I/O processing, it also supports XML/SOAP interface based on DPWS to ease up the integration of industrial processes in a SOA context. The control programs can be defined using the integrated browser-based editor supporting IEC61131-3 Structured Text language and configured to be triggered whenever a linked service operation is invoked. This equipment also supports the triggering of events within control programs.



(a) MOFA France kit (b) MOFA France components overview

Figure 6.9: MOFA France educational kit

Exploiting the distributed power allowed by this set of S1000, the control application was deployed among these devices in an approach following and testing the granularity guidelines described in section 3.3.

As the current solution relies on a standard DPWS stack, all devices and services are possible of being discovered using a standard device explorer. For initial testing purposes it was used the *DPWS Explorer* developed by [WS4D, 2012a] to check the current solution standard-compliancy and to dynamically test the implemented services.

6.2.2.2 Modelling

As for the skills concept in [Barata and Camarinha-Matos, 2003], in this context services represent the abilities or functionalities that characterise modules or production components. These services are the building blocks included in the process plan whenever there is a need to launch a production order. Due to the insufficient number of I/O available in each S1000 device, the *Crane* device had to be decomposed into two different devices: one device controlling the movement in the *X* axis and another controlling the movements in the axes *Y* and *Z*. This approached revealed as good test bench for an orchestrator solution running on a regular PC using DPWS Java stack by [WS4D, 2012b]. In the network, this new *Crane* device will appear as any other device deployed on a S1000 and will coordinate both devices responsible for the axes control to enact a more complex service: *PickAndPlace*. This service materializes as a predetermined sequence of invocations to the services hosted by the *AxisX* and *AxesYZ* devices. Fig. 6.10 presents the UML Sequence

diagram of the messages exchange between these devices to perform a *PickAndPlace* operation. In this last diagram it is possible to detect several kinds of MEPs. Regarding atomic services, i.e. services that are not composed or dependent of services hosted by other devices, the approach followed was to provide an instantaneous reply for every invocation received.

The WSDL of *moveAxisX* service is show in Listing 6.4 as example. In this simple atomic service WSDL it is possible to identify both the request-reply *moveX* and event *reachedX* operations.

Listing 6.4: *moveAxisX* WSDL example

```
<?xml version="1.0" encoding="UTF-8" ?>
<definitions name="moveAxisX"
targetNamespace="http://www.uninova.pt/wsdl/moveAxisX" ... >

    <types>
        <xsd:schema targetNamespace="http://www.uninova.pt/wsdl/moveAxisX"
            elementFormDefault="qualified">
            <xsd:element name="moveXResponse">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="coordXResponse" type="xsd:string"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="moveX">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="coordX" type="xsd:string"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:complexType name="reachedXType">
                <xsd:sequence>
                    <xsd:element name="reach" type="xsd:string"/>
                </xsd:sequence>
            </xsd:complexType>
            <xsd:element name="reachedX" type="tns:reachedXType"/>
        </xsd:schema>
    </types>

    <message name="moveXRequestMsg">
        <part name="body" element="tns:moveX"/>
    </message>
    <message name="moveXResponseMsg">
        <part name="body" element="tns:moveXResponse"/>
    </message>
    <message name="reachedXMsg">
        <part name="body" element="tns:reachedX"/>
    </message>
```

```
40
41      <portType name="moveAxisXServicePortType" wse:EventSource="true">
42          <operation name="moveX">
43              <input message="tns:moveXRequestMsg"/>
44              <output message="tns:moveXResponseMsg"/>
45          </operation>
46          <operation name="reachedX">
47              <output message="tns:reachedXMsg"/>
48          </operation>
49      </portType>
50
51      <binding name="moveAxisXServiceBinding"
52      type="tns:moveAxisXServicePortType">
53          <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
54          style="document" />
55          <operation name="moveX">
56              <soap:operation style="document" />
57              <wsdl:input>
58                  <soap:body use="literal" />
59              </wsdl:input>
60              <wsdl:output>
61                  <soap:body use="literal" />
62              </wsdl:output>
63          </operation>
64          <operation name="reachedX">
65              <soap:operation style="document" />
66              <wsdl:output>
67                  <soap:body use="literal" />
68              </wsdl:output>
69          </operation>
70      </binding>
71
72      <service name="moveAxisX">
73      </service>
74  </definitions>
```

This acknowledgement message indicates that the invocation was well received and the appropriate action will be executed as soon as possible. Later on, an event is triggered to signalize the result of the previous invocation. In relation to an orchestrator device, such as the *Crane* that hosts *CraneService*, it was employed a more classical approach of synchronous Request-Reply as in 6.10.

A similar solution was implemented for the *PaintAndWeld* that orchestrates the *Conveyor* and both *Weld* and *Paint* workstations to support a composed service that executes the ampler process of "welding" and/or "painting" encompassing the according transport by the conveyor belt. The difference in this case is that the orchestrator runs in a S1000 being implemented using Structured Text code that invokes each service in the predetermined flow.

These two solutions support the argument that after services being available in the

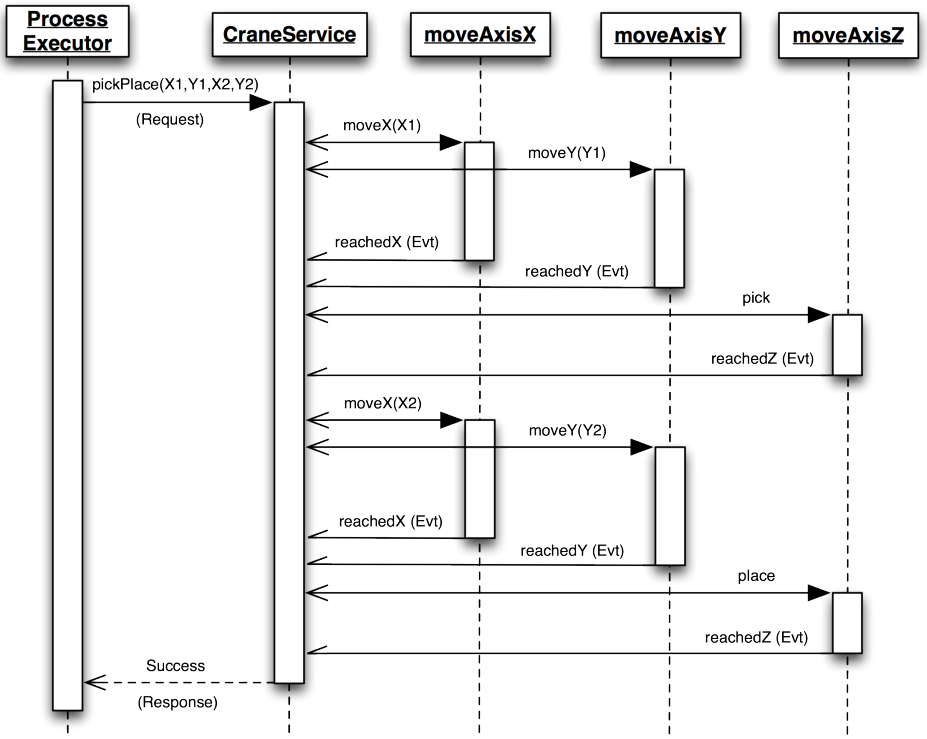


Figure 6.10: Crane Pick and Place example – UML sequence diagram

network to be discovered and employed, the systems integrator can chose the coordination solution that best fits current system requirements. In this context, the system integrator avoids low level implementations and simply focuses on creating the desired system behaviour based on a composition of available services. A complete overview of the deployed distributed service-oriented control solution for the MOFA France kit is depicted in Fig. 6.11.

6.2.2.3 Variations in Service granularity

While designing service interfaces for each component of the MOFA environment different levels of interface granularity were tested. For the *Dry* workstation instead of creating a service that encapsulates the overall process of "drying", as for the *Drill* workstation, the author decided to offer services with a thinner granularity: *dryMachine*, *translationTable* and *toolsCollection*. This decision implies that the systems integrator when designing a process plan needs to know that in order to perform the *Dry* task as a whole there is a need to move the translation table, chose the appropriate drying tool, trigger a dry operation at the drying machine, and when it is finished then move the translation table again so that the pallet can be transported elsewhere by the crane. Although being a valid option, this solution revealed to be frustrating when composing the process plans at the Process Management Tool since it assumed the user to know more details on the

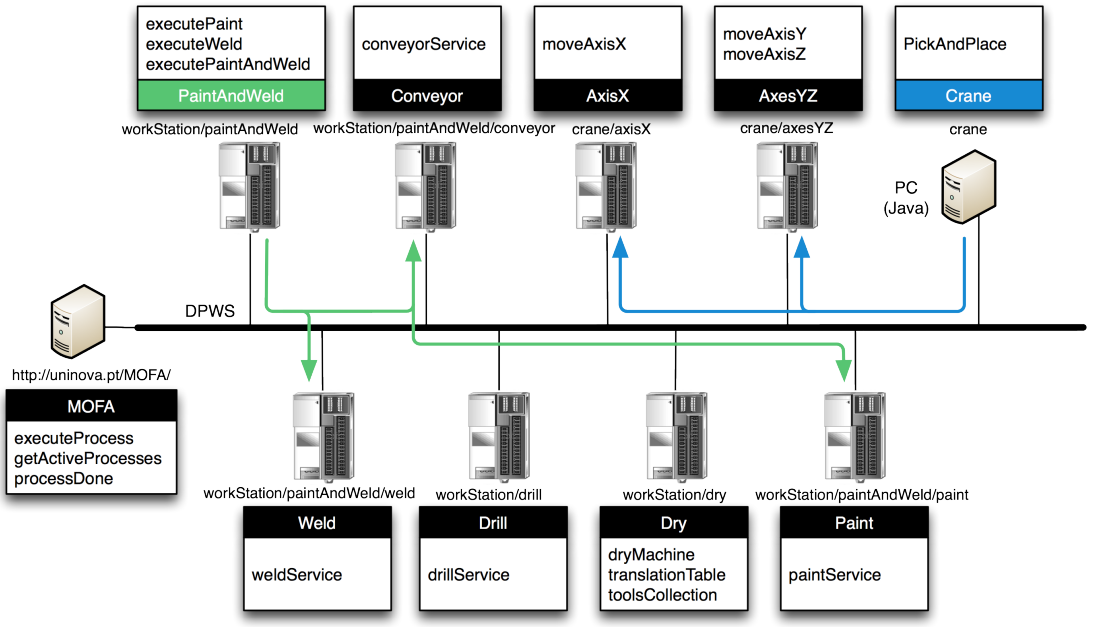


Figure 6.11: MOFA France distributed service-oriented control architecture

dry process than expected. In this context, the user wants to create his process plan by combining courser-grained processes possible of being executed to the pallet instead of having to deal with local subprocess tasks.

An example of the displayed information on a standard device explorer for the *DryWorkstation* is shown in Fig. 6.12. In Fig. 6.12(a) it is possible to visualize the device, hosted services and operations for *DryWorkstation* in a tree-map view. In Fig. 6.12(b) the device metadata is shown for that same device.

6.2.2.4 Architecture Elements

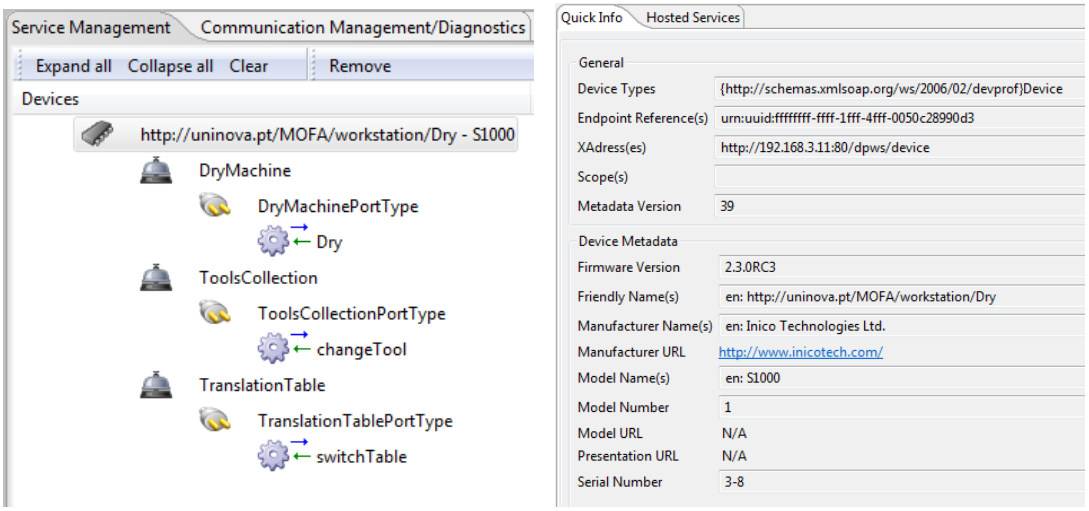
This section details the implementation work done in the scope of each architecture component as described in section 4.2.

6.2.2.4.1 Device Explorer

As referred before, this element is a software component mostly used during analysis, setup and troubleshooting of the service-oriented application. It comprises a perceptive GUI (see Fig. 6.13) to allow a more effortless and straightforward execution of the supported tasks. The next subsection will detail each process supported by the implemented Device Explorer prototype

6.2.2.4.1.1 Retrieval of Logical Topology

Besides discovering and retrieving metadata from the devices available in the network whenever the user clicks the *Search* button (upper left corner in Fig. 6.13), the prototyped



(a) DryWorkstation detail: device, services and operations

(b) DryWorkstation metadata

Figure 6.12: Device details using a standard DPWS explorer

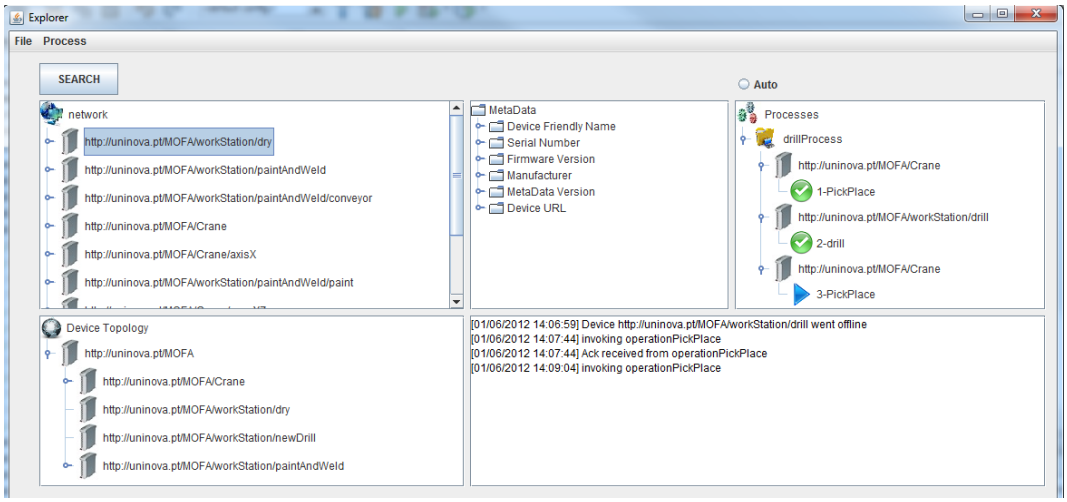
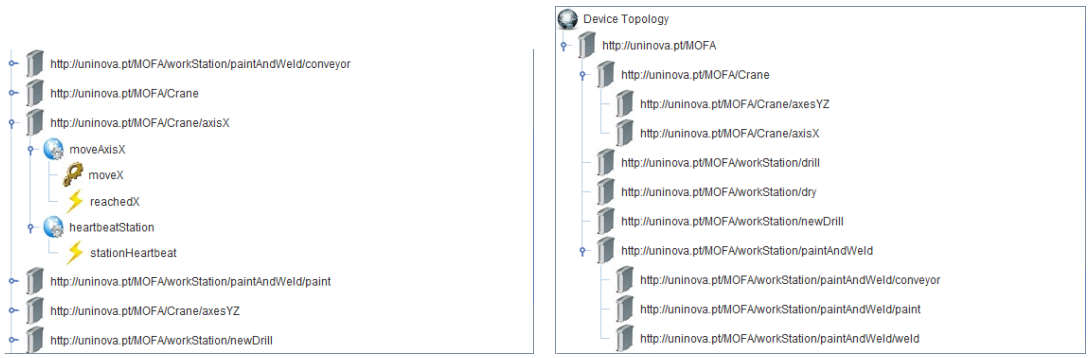


Figure 6.13: Detail of Device Explorer prototype GUI



(a) MOFA flat network (b) MOFA device topology

Figure 6.14: MOFA network and device topology visualization

Device Explorer developed in the scope of this work also extracts and presents the current device topology, i.e. how devices are logically composed only based on local information exchange. This way the systems integrator can retrieve details about an existing device, such as its metadata, hosted services, operations and event sources in addition to the visualization of devices by layers of abstraction.

Fig. 6.14(a) shows a list of some of the MOFA devices available in the network along with its hosted services and their embraced operations and events. *moveX* is an operation and *reachedX* is an event source supported by the *moveAxisX* service hosted by the device *AxisX* which is responsible for controlling the movement on the "x" axis of the crane.

As previously shown on Fig. 6.10, this device in particular is used by the *Crane* orchestrator along with the *AxesYZ* device that controls the movements of the "y" and "z" axes to perform a *PickAndPlace* operation. Just by looking to the list of available devices it would be complicated to determine who controls whom, or who exposes coarser-grained services based on atomic services provided by other devices.

As presented in section 4.3.3, the URI of each device *FriendlyName* will reveal its position on current device topology. By following this guideline to define each device *FriendlyName* it was possible to determine current device topology only based on local information available on each device metadata.

In Fig. 6.14(b) it is also visible that *Crane* device orchestrates *AxisX* and *AxisYZ* devices. This feature will offer systems integrator an improved overview of the current system topology only based on local information held by each device to better assess the implications of modifying a part of the system or determine a fault origin.

6.2.2.4.1.2 Heartbeating monitoring

To support this capability, a heartbeat service was implement is each device that triggers a *heartbeating* event in predetermined cadence. Whenever the Device Explorer discovers a device, it subscribes to the hosted by default *heartbeatService* and start to receive and monitor the arrival of its events. Whenever a device goes offline for some reason, i.e.

connection problems, hardware conflicts or a power surge, Device Explorer is able to update the status of the device with the information provided by the event or by the lack of it. Hence, after subscribing to a *heartbeatService* event, a control thread is associated to it and launched to monitor every change of the status of one device. This control thread activates a timer, which expires after a configurable time. When an event is triggered, it refreshes its own timer, resetting it to the configured maximum time interval. Consequently, when this timer expires it means that the device is no longer available in the network and Device Explorer will trigger a warning message.

This solution revealed to increase the robustness of the architecture and awareness of the system status by early detecting faulty devices and reduce the time of response to anomalies.

6.2.2.4.1.3 Other features

Besides the functionality already described in previous subsections, the Device Explorer prototype also presents the device metadata of a selected device (upper middle frame in Fig. 6.13) in a tree map format. Since the current prototype is deeply integrated with Process Management Tool, it is possible to monitor the execution of production processes directly in the Device Explorer GUI (upper right frame in Fig. 6.13).

More details about the Process Management Tool prototype are available in section 6.2.2.4.2. Also, the Device Explorer is integrated with the Semantic Assistant to deliver both semantic translation and semantic gateway deployment solutions as further detailed in section 6.2.2.4.3.

6.2.2.4.2 Process Management Tools

In the scope of the validation of this thesis, a simple Process Management tool prototype was developed in order to allow the execution of production processes at the MOFA kit. Each process plan would correspond to a workflow of "operations" needed to be performed on a particular kind of "material" (simulated by a wood pallet) to be transformed into a "product part". The focus was not concretely on providing an advanced process management environment but to allow the creation of some simple processes that employed services hosted by the available devices. The goal was then to verify the different approaches proposed in this thesis to cope with device level variations.

6.2.2.4.2.1 Process Executor Engine

As seen in Fig. 6.15 the user can pick the operations directly from the list of devices discovered in the network, enter the according parameters and compose them to create a sequential process plan. This process is saved in a XML format as shown in Listing 6.5 so that it can be reused in posterior executions. When a systems integrator decides to launch a new process instance he can retrieve one previously stored or create a new one from scratch and start its execution for a particular pallet. The tool will then trigger a new

generic process execution thread that will be responsible for executing and monitoring the correct execution of the process plan.

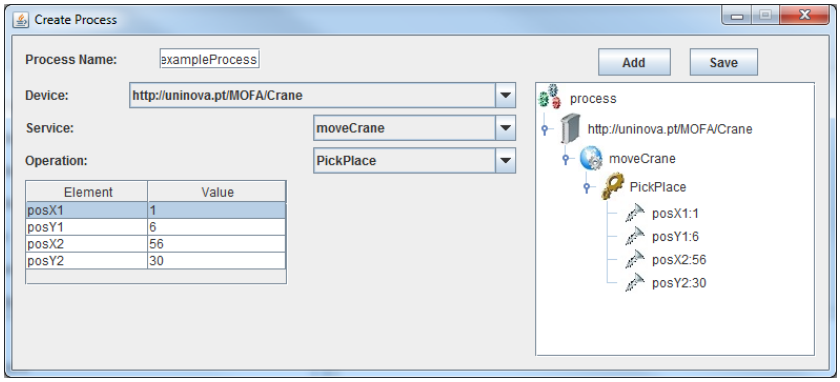


Figure 6.15: Creation of Process Plan GUI

Listing 6.5: Outline of a Process Plan in XML format

```
<xmlProcess name="drillProcess">
   <device name="http://uninova.pt/MOFA/Crane">
      <service name="craneService">
         <operation name="pickPlace">
            <input name="posX1">
               <value>1</value>
            </input>
            <input name="posY1">
               <value>6</value>
            </input>
            <input name="posX2">
               <value>56</value>
            </input>
            <input name="posY2">
               <value>30</value>
            </input>
         </operation>
      </service>
   </device>
   <device name="http://uninova.pt/MOFA/workStation/Drill">
      <service name="drillService">
         <operation name="drill">
            <input name="DrillTime">
               <value>3000</value>
            </input>
         </operation>
      </service>
   </device>
```

Once a process is running, it is possible to follow the status of each plan operation: if it is being executed, already finished with success or waiting to be invoked (see Fig. 6.16).

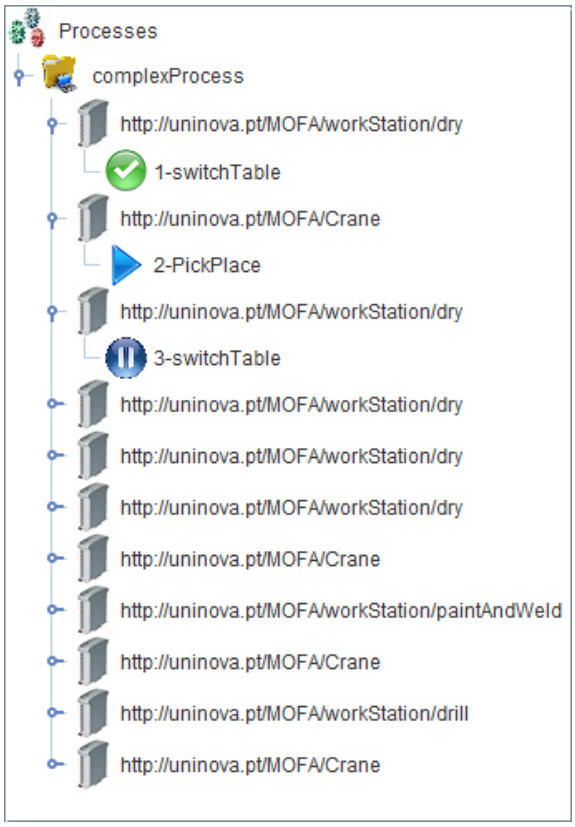


Figure 6.16: Monitoring GUI frame for an active Production Process

This generic process executor engine was also implemented to support the concurrent execution of multiple processes. For this reason, each time a process is loaded, a new thread is launched and the according process detail will be added to a new view tree. This feature enabled a better supervision of production processes running in the MOFA system.

6.2.2.4.2.2 Services Transparency

As expected due to the intrinsic abstraction layer provided by the services interfaces and WS-Discovery peer-to-peer mechanism included in DPWS middleware, the exchange of a device by another presenting the same interface is done effortlessly without any programming or configuration involved.

To test this feature, a PLC responsible for the control of a MOFA workstation was disconnected (powered off and unplugged I/O connections) and a new one exposing similar metadata and identical service interfaces was put in place. This transition revealed no issues or any other kind of glitches – the active process run as expected using these "new" services. When launching a production process, the Process Management tool searches the network for all the required service operations to verify if it is able to execute the whole process before launching it. If during run-time a device and consequently its hosted service become unavailable, the process task is able to handle an invocation error

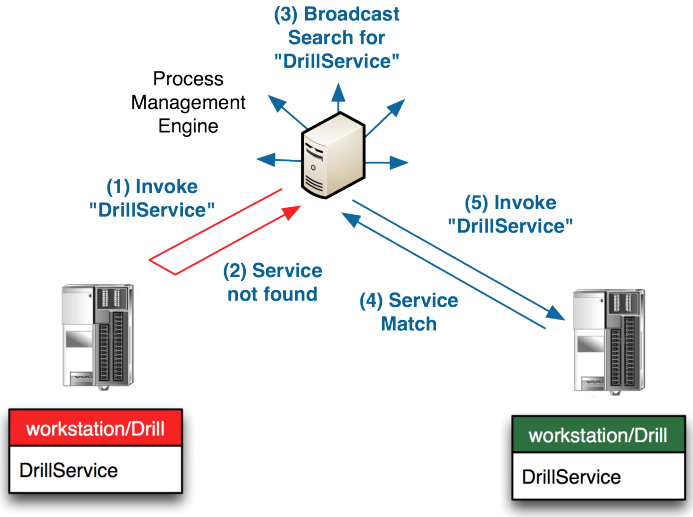


Figure 6.17: Transparent exchange of devices hosting identical services – *DrillService* example

due to an out-dated reference, search the network for an identical service and continue the process as planned (see Fig. 6.17). The process executor will broadcast the network for a device with the same metadata and a hosted service with an identical interface as the one now missing. The discovery and invocation process is then completely transparent: no IP reconfigurations, no coding or any kind of reconfiguration of process plan – the systems integrator is simply notified about that "missing" device.

6.2.2.4.3 Semantic Assistant

Although being available as independent element exposing its services in the network, as enunciated before, the Semantic Assistant was integrated with the Device Explorer to deliver the user a richer graphical interface when executing these semantic-related tasks. The use and relevance of these are then presented in the scope of process definition, modification or when coping with a faulty device or service.

6.2.2.4.3.1 Semantic Matching

In a situation where it is not feasible to plug a device with an identical metadata and hosted services of a device that was unplugged for maintenance or inactive due to a fault, an alternative solution must be available. In these conditions the system needs to quickly adapt by assisting the systems integrator or even provide an automated resolution.

For the case when an identical "empty" device is available to substitute a missing one, the human simply needs to retrieve previous configuration from Service Store and use a Service Deployer to redeploy the correct application. Of course, depending on system autonomy and security constraints, this task can be automated using a FDR mechanism.

This particular case must take into account the type of device and service while checking the implications of substituting that particular device during system run-time. This particular solution was not implemented on current test-bench.

When this solution is not feasible or when it is important to ensure an immediate solution while the faulty device is not substituted, the present implementation can also provide means to handle it. After identifying the missing or faulty components, the Semantic Assistant is solicited to retrieve in the network services semantically equivalent to those now currently unavailable. Each operation is specified through a OWL-S file so that each operation is semantically defined in terms of profile, functionality, parameters and grounding. The prototype tool is able to create automatically a OWL-S file for every existing operation by extracting information from its service WSDL and device metadata. Whenever an operation is found in the network, an OWL-S file is generated with WSDL2OWLS tool provided by OWL-S API [Manchester University, 2012]. This tool is used to create the basic structure of an OWL-S file with the help of device metadata and service description.

By having this information stored and updated, it was possible to run reasoning tasks to determine which services can be equivalent to the one missing. The user can also provide some input on these files to better specify some of the features required in the matching process. It is always easier to update some parameters on a OWL-S file than reprogram a new device or reconfigure a process plan every time a device goes offline using one of the available ontology editors.

6.2.2.4.3.2 Semantic Translation

In an extended version of the previous subsection scenario, the process engine when executing a process plan might not found any device or service identical to the one missing. Whenever this happens the system will try to discover a device semantically similar hosting a compatible operation and provide a translation mapping. While discovery implies service description and semantic tags comparison, the invocation might need some translation mechanism to adapt to a possible unlike interface – services coming from different sources might differ in the interface although implementing the exact essential function.

The applied approach was to use this mapping to invoke this equivalent service instead of the missing one while executing the current process plan. Despite the existence of an OWL-S file for every discovered service, there is still a need to know how it is going to be possible to translate from a service to a semantically equivalent one.

A GUI was implemented to assist the user when creating a semantic translation between two services with different interfaces (see Fig. 6.18). It shows all available OWL-S files and converts them to a tree map with device, service, operation and inputs representation using RDF Data Query Language (RDQL) queries. By selecting one of the operations in the left frame, this same service operation can be mapped to another one. The user simply needs to fill in the information related to the new service and how the

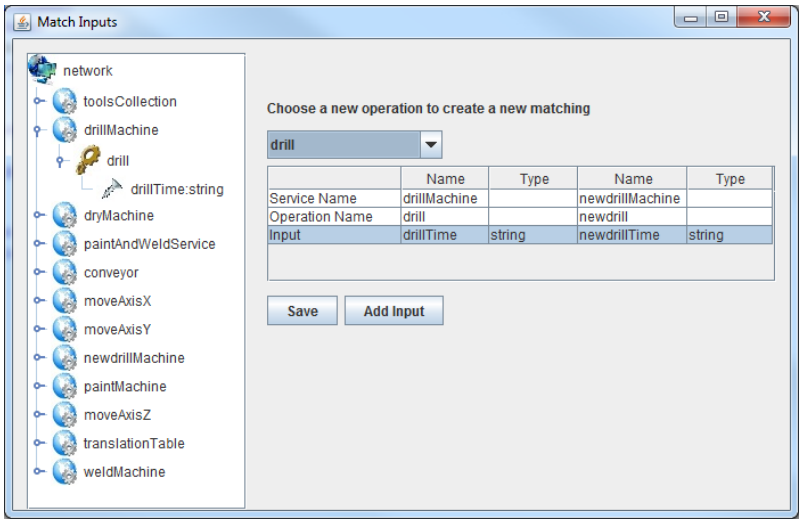


Figure 6.18: Semantic mapping of service interfaces

inputs should be mapped between them. This translation is also saved into a XML format as shown in Listing 6.6 to cope with future translation needs. This way, the process executor can now map a missing service to an equivalent one using this same mapping and proceed with the process executing by relaying on the data entered by the systems integrator and not discarding the current process just because the original service is not available at that moment. The current prototype is also able to handle the matching of operations where the number of inputs is not same in both operations. In these situations, the user can specify which inputs should not be taken into account when performing the translation.

This decision can be mediated by the systems integrator or automated for the use-cases that do not involve critical resources or raise security concerns (*auto* switch on top right in Fig. 6.13).

Listing 6.6: Outline of a Semantic mapping in XML format

```
<matching>
  <serviceMatched>
    <serviceName>drillMachine</serviceName>
    <operationName>drill</operationName>
    <serviceNameMatched>newdrillMachine</serviceNameMatched>
    <operationNameMatched>newdrill</operationNameMatched>
    <inputType>
      <name>drillTime</name>
      <nameMatched>newdrillTime</nameMatched>
      <xsdtype>string</xsdtype>
      <xsdtypeMatched>string</xsdtypeMatched>
    </inputType>
    <newInput>
      <inputName>invalid</inputName>
      <inputValue>invalid</inputValue>
```

```
16      </newInput>
17    </serviceMatched>
18  </matching>
```

6.2.2.4.3.3 Semantic Gateway

The previous use case assumed that services were simply invoked by a process task and this last was responsible for retrieving the appropriate semantic translations whenever necessary. However, some other network elements might also require the services hosted by that currently inexistent service. In this situation, the deployment of a semantic gateway would assume an essential role on emulating previous device interfaces. This semantic gateway will copy the interface of the missing device and hosted services and translate any invocation that it receives to the equivalent service interface. See Fig. 4.14 in section 4.3.7 for an example. For the device that was consuming the missing service, it will now discover the semantic gateway as if it were the original device and invoke its services again as if it was the previous one – the device metadata and the hosted services interface are the same.

In the current application, using the Device Explorer GUI, the systems integrator can detect or be informed when a device is currently unavailable and create a semantic gateway simply by choosing the according previously stored OWL-S definition. The Semantic Gateway GUI (see Fig. 6.19) will then launch a DPWS device which clones the device metadata and service interfaces of the inexistent one. This new device is able to handle incoming communications (following the original missing service interface), retrieve the available semantically equivalent by exploiting the Semantic Assistant services and execute the translation of interface and parameters so that the equivalent service in now employed. From the invoker point-of-view, the execution of its process will proceed as before, i.e. as if the original device and service are still available.

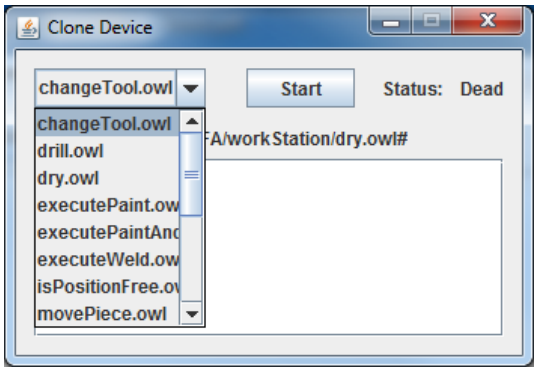


Figure 6.19: Creating a Semantic Gateway

6.2.2.5 Integration Tests

In order to ensure that all architecture elements were correctly integrated and operating as expected, several wide-range tests were specified having in mind a complex combination of processes and architecture elements to show the relevance of the current work. All tests had in common the simulation of a faulty device and need to update an existing process plan or device. The following test description and results summarize the overall functionality and performance of the proposed architecture.

Initially, three production plans were created using the prototyped Process Management Tool. The first process invokes a drill operation to be done to a pallet coming from the warehouse and consequent store in the same local, which includes the transport to the *Drill* workstation and return by invoking the *PickAndPlace* service provided by the *Crane* device. Similar to the first one, the second process instead requires the pallet to be dried. The difference here is that *Dry* workstation was modeled with a lower granularity which obliges the plan creator to know the correct sequence of actions to be performed to completely fulfill the dry procedure. This process also includes a *PickAndPlace* operation from and to the warehouse. The third process includes operations from all workstations. It starts by a move from the warehouse to the *Drill* workstation to be drilled, then a move to the *PaintAndWeld* where the conveyor belt will transport the pallet to be painted by the *Paint* device. From the end of the conveyor belt it will be moved to the *Dry* workstation where the paint will be dried. Finally, the pallet is moved back to the warehouse. All moves are again executed by exploiting the *PickAndPlace* operation from *Crane* device.

Prior to launching these three processes in the demo system, the PLC where *Drill* workstation and according *drillService* are deployed is abruptly removed from the network by unplugging its power supply. For testing purposes, a device hosting a service named *newDrillService* was also made available in the network although no reference to it exists in the three process plans presented before. From the processes plan point of view, only *drillService* is known.

All these processes were then simultaneously launched and executed in parallel. For this reason and since there is only a single physical crane able to transport pallet between the different stations, the first request to arrive to *Crane* will be promptly executed and the later ones are put in queue for posterior execution. In the present test, the firstly described process was also the first one the be executed but as described before, the *Drill* workstation is no longer available. The Process Management tool detects this abnormal situation and requests the Semantic Assistant to find, if possible, an equivalent service in order to continue the current production process. By precaution, a semantic mapping was set stating the mapping from the original *drillService* and *newDrillService* in order to provide extra robustness to the system, i.e. the system integrator had set this mapping rule to cope with these deviant situations. Due to this, the Semantic Assistance will propose the *newDrillService* as a replacement for the missing one. The process executor will handle this information and it will invoke the existent equivalent service instead of the

one predefined in the original XML process file. The decision to take the replacement suggested by the Semantic Assistant can be automatic or through user notification and consequent validation or refusal of the proposal. In the current test, this decision was set to automatic, i.e. accept whatever proposal the Semantic Assistant provides. This service replacement process is taking place while other tasks are still being executed in the shop floor as long as they do not interfere with the *Drill* workstation.

Assuming that the original *drillService* will be unavailable for an indeterminate period of time, the systems integrator decides to deploy a Semantic Gateway to ease the upcoming interactions with the inexistent device. By using the Device Explorer, it was possible to launch the deployment of a Semantic Gateway by choosing which OWL-S file to use and create a *Drill* clone device exposing the same *drillService*. As referred before, every time a device is discovered in the network, an according OWL-S for its services is automatically generated. The user choses this same file to emulate the inexistent device (metadata and hosted services interfaces) and use the available semantic mapping to convey the invocations to the equivalent *newDrillService*. This solution is transparent for the service invokers since from the logical point of view it is the same device and services due to the augmented level of communication abstraction – detached from IP or other network configuration details. Although this particular solution involves an extra communications cost both in terms of bandwidth and round-trip delay plus some other security concerns, it proved to deliver an immediate solution while the original device is not replaced, repaired or finished maintenance.

In this test, all three production processes where then successfully accomplished. Inclusively, the third process that also included an invocation to the original *drillService*, transparently employed the cloned services offered by semantic gateway deployed while it was waiting to be transported by the *Crane* device.

Regarding granularity options, during the tests phase coarser granularity revealed to be the preferred option since when creating a process plan the user wants to set a particular sequence of activities that do not involve a deep knowledge of the process executed by each workstation. From the process point of view, the granularity should stay at the workstation level, i.e. for each workstation, which operations is it able to execute as responding to the question: *"what can this workstation do to my pallet?"* A finer granularity must be only adopted for troubleshooting and diagnosis purposes.

6.3 Ongoing Developments

6.3.1 Merging OPC UA & DPWS

Driven by the envisaged benefits coming from a combination of both specifications plus other complementary WS-* standards such as WS-Management, a first effort was framed under the scope of SOCRADES [Mensch and Jammes, 2008]. This combined solution promoted mutual interoperability and paved the way for an enhanced industrial automation

device level interoperability compliant with SOA principles and technology.

Schneider Electric's open source implementations of DPWS and WS-Management available at [SOA4D, 2009a] can provide the foundations for an integration solution. Both stacks are open and extensible enough to support the addition of the required security protocols and optimized transport layers, together with the development tools necessary to ease the implementation of the OPC UA web services on top of DPWS.

Even if a OPC UA application could be implemented in embedded devices, the required memory footprint is still most of the times too high for very low cost and low power devices. Defining a low memory footprint profile combining the DPWS and WS-Management protocols and a subset of the OPC UA data model implemented in C would be a very attractive solution, which could be endorsed in the future by the OPC Foundation.

Although a concrete solution has been proposed, due to some patent rights issues between some of the project partners it was temporarily put on hold. However, the merge of these two specifications is foreseen by extracting the best that each of them endorse. It promises to deliver a more complete and adaptive specification that will act as an essential SOA instrument to be employed at device level in the industrial automation domain.

The prominence of this merging solution presented in this document is stated by all subsequent work that incorporated it such as [Bony et al., 2011, Izaguirre et al., 2011, Kyusakov et al., 2011, Seilonen et al., 2011, Son and Yi, 2012]. Besides these, SOCRADES follow-up project AESOP [AESOP, 2012] is expected to deliver a final demonstrator implementing a merging OPC UA/DPWS solution based on the current work.

6.3.2 Projects Follow-up

The first-known application of SOA at device level in industrial automation was accomplished during SIRENA project [SIRENA, 2005] as documented in [Jammes and Smit, 2005a]. The follow-up projects such as SODA [SODA, 2009], SOCRADES [SOCRADES, 2009], InLife [InLife, 2008] and RI-MACS [RI-MACS, 2012] pushed the community and major industrial players to extend the R&D effort on this topic and expand the range of solutions and domains of application. The present work was framed in part by these projects and the author exploited these consortiums know-how to discuss, share and delineate the solution presented on this document. Also, some parts of this work were integrated into some of the projects deliverables and results. The final evaluation of SODA project, on which the present work including one of the demonstrators was integrated, revealed the major relevance of the work to the domain. Inclusive, SODA project has been awarded the ITEA Achievement Award – Bronze for outstanding contributions to the ITEA program.

Although much progress as been accomplished since SIRENA days (also a ITEA

Achievement Award – Gold recipient project), there is still a crescent interest on the investigation and application of SOA solutions into several industrial branches as corroborated by the large number of international R&D also investing topics within the SOA for industrial automation framework (see Table 6.4). These projects can use the present work as an input for pursuing more advanced solutions and implement more extensive and complex tests and benchmarks.

With a big number of initiatives interested to pursue the research of SOA related aspects and applications to several industrial areas, it becomes evident that this topic is currently relevant to the industry community and it is expected to continue to expand and deliver valuable contributions to the domain.

6.4 Scientific Contributions and Peer Validation

6.4.1 Results from the present work

The work developed in the framework of this thesis was also peer reviewed in several international publications. The following research works were submitted and accepted:

- **International Journals**
  - Cândido, G., Colombo, A., Barata, J., and Jammes, F. (2011). Service-oriented infrastructure to support the deployment of evolvable production systems. *Industrial Informatics, IEEE Transactions on*, 7(4):759–767. IEEE.
  - Cândido, G., Barata, J., Colombo, A., and Jammes, F. (2009). SOA in reconfigurable supply chains: A research roadmap. *Engineering Applications of Artificial Intelligence*, 22(6):939–949. Elsevier.
- **Conferences with peer reviewing**
  - Cândido, G., Sousa, C., Di Orio, G., Barata, J., and Colombo, A. (2013). Enhancing Device Exchange Agility in Service-oriented Industrial Automation. In *IEEE International Symposium on Industrial Electronics*, Taipei, Taiwan, May 28-31, 2013.
  - Cândido, G., Barata, J., and Colombo, A. (2012). Service-oriented Infrastructure at Device Level to Implement Agile Factories. In *International Conference on Systems, Man and Cybernetics*, pages 1171–1176. IEEE.
  - Ribeiro, L., Cândido, G., Barata, J., Schuetz, S., and Hofmann, A. (2011). IT support of mechatronic networks: A brief survey. In *Industrial Electronics, International Symposium on*, pages 1791–1796. IEEE.
  - Cândido, G., Jammes, F., Barata, J., and Colombo, A. (2010). SOA at device level in the industrial domain: Assessment of OPC UA and DPWS specifications. In *Industrial Informatics, International Conference on*, pages 598–603. IEEE.

Project Name	Short Summary
AESOP [AESOP, 2012]	Investigation of a SOA approach for monitoring and control of very large scale Process Control Systems (batch and continuous process applications). Development of a SOA-based approach for next generation of SCADA/Distributed Control System (DCS) systems targeting Process Control Applications.
IoT@Work [IoT@Work, 2012]	Development of self-configuration mechanisms, enabling what we call secure plug and work Internet of Things (IoT), whereby devices are auto-configured and ready to co-operate with each other as soon as they are plugged into the factory network, self-adapting to changes in response to demands, faults, etc.
eSonia [eSonia, 2012]	Realization of a asset-aware and self-recovery plant through pervasive heterogeneous IPv6-based embedded devices, promoting on-board specialised services and *glue* all elements through a middleware capitalising a SOA approach
iProd [iProd, 2012]	Improvement of the efficiency and quality of the Product Development Process developing a flexible, service oriented, customer driven software framework. To achieve these goals, iProd will rely on knowledge management (KM), knowledge based engineering (KBE) and process integration and automation technologies.
COSMOS [COSMOS, 2012]	Design, development and implementation of a control system for factory management by implementing a flexible, modular and evolvable automation approach. Focus on wind turbine assembly process.
EPES [EPES, 2012]	Aims to develop a set of ICT tools to support the easy configuration/adaptation of new services, storing and reuse of the apprehended knowledge in order to improve the services and enable the continuous improvement of products along its life cycle by applying best up to date technologies.
BEMO-COFRA [BEMO-COFRA, 2012]	Supported by the results of Hydra IP, Pobicos and ebbits IP projects featuring a SOA and a middleware able to expose smart objects, legacy devices and subsystems capabilities by means of web services, BEMO-COFRA will address both technological aspects and user needs to promote a wider adoption of large-scale networked monitoring and control solutions.
ExtremeFactories [ExtremeFactories, 2012]	Creation of a new methodology for accelerating the adoption of innovation processes in Small/Medium Enterprise (SME)s by designing and developing an internet-based platform, with semantic capabilities, that will implement as services the concepts of the methodology.
Self-Learning [Self-Learning, 2012]	Development of a highly reliable and secure service-based self-learning and self-optimization solutions aiming at a tight integration of control & maintenance of production systems.
MSEE [MSEE, 2012]	Make Service Science Management and Engineering (SSME) evolve from a methodological viewpoint to adapt, modify, extend SSME concepts so that they could be applicable to traditionally product-oriented enterprises; from an implementation viewpoint to instantiate service oriented architectures and platforms for global manufacturing service systems.
PREMANUS [PreManus, 2012]	Overcome the asymmetric distribution of information in the End-of-Life recovery of products by connecting OEMs and subcontractors. To achieve this goal, PREMANUS promotes on demand middleware that combines product information and product services using SOA.
Industry 4.0 [GTAI, 2012]	Launched by German government, it is expected to open the industry sector to reference architectures to connect factories and global value networks enabling them to share data and to connect business process with SOA.
Arrowhead [Arrowhead, 2012]	Also launched by the Artemis initiative, the vision is to enable collaborative automation by networked embedded devices. This goal is expected to be achieved by enabling the interoperability and integration of services provided by almost any device.

Table 6.4: Ongoing international R&D projects and initiatives addressing SOA applications for industrial automation

- Cândido, G., Barata, J., Colombo, A., and Jammes, F. (2010). Service-Oriented Architecture at device level to support Evolvable Production Systems. In *Industrial Electronics, IEEE International Symposium*, pages 2669–2674. IEEE
- Cândido, G., Jammes, F., Barata, J., and Colombo, A. (2010). Semantic SOA approach to support agile reengineering at device level. In *Workshop on Intelligent Manufacturing Systems*. IFAC.
- Cândido, G., Jammes, F., Barata, J., and Colombo, A. (2010). Applications of Dynamic Deployment of Services in Industrial Automation. In *Emerging Trends in Technological Innovation – Proceedings of IFIP WG 5.5/SOCOLNET Doctoral Conference on Computing, Electrical and Industrial Systems*, volume 314 of IFIP Advances in Information and Communication Technology, pages 151–158. Springer.
- Ribeiro, L., Barata, J., Cândido, G., and Onori, M. (2010). Evolvable Production Systems: An Integrated View on Recent Developments. In *CIRP-Sponsored International Conference on Digital Enterprise Technology*, pages 841–854. Springer.
- Cândido, G., Jammes, F., Barata, J., and Colombo, A. (2010). Generic Management Services for DPWS-enabled devices. In *IEEE Industrial Electronics Society, Annual Conference of the*, pages 3931–3936. IEEE.

6.4.2 Other Applications of SOA

In parallel to the present work the author also participated in other collaborative R&D projects such as FP7 NMP Self-Learning and ITEA NEMO&CODED projects which employed SOA paradigm into new realms of application and validated its application into domains far from the original business ICT domain. In the first case, SOA concepts and methodologies were applied to support the optimization of production equipment and systems performance based on intelligent context extraction and adaption assisted by machine learning techniques. In the latter case, SOA technology was deployed into smart devices working in the context of electrical production, distribution and consumption envisioning improved energy efficiency by enabling distributed monitoring, diagnostics and control. In the scope of these projects, the author participated in the following contributions:

- **Conferences with peer reviewing**:
  - Di Orio, G., Cândido, G., Barata, J., & Bittencourt J., Bonefeld R. (2013). Energy Efficiency in Machine Tool – A Self-Learning Approach. In *IEEE International Conference on Systems, Man, and Cybernetics*, Manchester, UK, October 13-16, 2013.
  - Di Orio, G., Cândido, G., Barata, J., Scholze, S., Kotte, O. (2013). Self-Learning

Production Systems (SLPS) – Self-Learning approach to support lifecycle optimization of Manufacturing. In *Annual Conference of the IEEE Industrial Electronics Society (IECON 2013)*, Vienna, Austria, November 11-13, 2013.

- Di Orio, G., Cândido, G., Barata, J., & Scholze, S., Kotte, O., Stokic D. (2013). Self-Learning Production Systems (SLPS) - Optimization of Manufacturing process parameters for the Shoe Industry. In *IEEE International Conference on Industrial Informatics (INDIN 2013)*, pages 386–391, Bochum, Germany, July 29-31, 2013.
- Cândido, G., Di Orio, G., Barata, J. (2013). Self-Learning Production Systems (SLPS) – Adapter Reference Architecture. In *International Conference on Flexible Automation and Intelligent Manufacturing (FAIM)*, pages 681–693, Porto, Portugal, June 26-28, 2013.
- Cândido, G., Di Orio, G., Barata, J., Bittencourt, J., Bonefeld, R. (2013). Self-Learning Production Systems (SLPS) – Energy Management Application for Machine Tools. In *IEEE International Symposium on Industrial Electronics*, Taipei, Taiwan, May 28-31, 2013.
- Cândido, G., Di Orio, G., Barata, J., and Scholze, S. (2012). Adapter for Self-Learning Production Systems. In *Technological Innovation for Value Creation – IFIP Advances in Information and Communication Technology*, 372:171–178. Springer.
- Uddin, M., Dvoryanchikova, A., Lastra, J., Scholze, S., Stokic, D., Cândido, G., and Barata, J. (2011). Service oriented computing to Self-Learning production system. In *Industrial Informatics, International Conference on*, pages 212–217. IEEE.
- Lima, J., Lima, C., Gomes, V., Martins, J., Barata, J., Ribeiro, L., and Cândido, G. (2011). DPWS as Specific Communication Service Mapping for IEC 61850. In *Industrial Informatics, International Conference on*, pages 193–198. IEEE.
- Lima, C., Gomes, V., Lima, J., Martins, J., Barata, J., Ribeiro, L., and Cândido, G. (2011). A standard-based software infrastructure to support energy efficiency using renewable energy sources. In *Industrial Electronics, International Symposium on*, pages 1175–1180. IEEE.
- Lima, C., Martins, J., Barata, J., Ribeiro, L., and Cândido, G. (2010). Towards a service based infrastructure to improve efficiency into energy systems: the NEMO&CODED quest. In *Workshop on Intelligent Manufacturing Systems*. IFAC.

6.4.3 The MAS background

The present work is also the result of a continuous interest in topics related with intelligent distributed applications for the industrial automation domain initiated during the author's participation in FP6 EUPASS project and elaboration of the Electrical and

Computers Engineering license final project. These are the contributions related to that background work in the domain of MAS applications for innovative shop floor control:

- **International Journal**:
  - Barata, J., Camarinha-Matos, L., and Cândido, G. (2008). A multiagent-based control system applied to an educational shop floor. *Robotics and Computer-Integrated Manufacturing*, 24(5):597–605. Elsevier.
- **Conferences with peer revision**
  - Barata, J., Cândido, G., and Colombo, A. (2007). A Multiagent based control system for an assembly cell. In *Workshop on Intelligent Manufacturing Systems*, volume 8, pages 116–121. IFAC.
  - Cândido, G. and Barata, J. (2007). A multiagent control system for shop floor assembly. In *International Conference on Industrial Applications of Holonic and Multi-Agent Systems: Holonic and Multi-Agent systems for Manufacturing*, volume 4659, pages 293–302. Springer.
  - Barata, J., Cândido, G., and Feijão, F. (2006). A multiagent-based control system applied to an educational shop floor. In *Information technology for balanced manufacturing systems: IFIP TC 5, WG 5.5 Seventh International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services*, volume 220, pages 119–128. Springer Verlag.

7

Conclusions

7.1 Summary of Research Challenges

The introduction of SOA into the domain of industrial automation as well as in other domains promised to deliver an important contribution over more conventional approaches while at the same time it raised new challenges and opportunities to domain experts.

- The industrial automation is characterized by a multi-protocol, -interface and -layer heterogeneity which has an immediate impact on overall interoperability and compromise system agility to face revisions of requirements.

- SOA approach was initially envisioned for business and enterprise level ICT and cannot be directly transferred to industrial automation device level without taking into account domain specificity.

- The industrial automation community is still suspicious about the introduction of contemporary ICT paradigms, particularly on what concerns semantic web or even Internet-based approaches, in a domain renowned by its critical real-time and safety constraints in communications.

- There are no *de facto* standards for service-oriented device model for an industrial automation context, as well no available service-oriented architecture designed to support device lifecycle evolution focusing on systems integrator assistance and consequently enhance overall system agility during reengineering or maintenance interventions.

7.2 Assessment of Research Hypotheses

The challenges previously enunciated sustained the establishment of three distinct but interrelated research questions:

RQ.1 – What are the key requirements and features still required to be addressed at device level in service-oriented industrial automation context?

The trailed hypothesis for the above research question is:

H.1 – The result of the retrieval and assessment of the key requirements and features that still need to be addressed at the industrial automation device level will be enriched if both SOA and industrial automation research vectors are combined and inferred which SOA concepts, principles and technology better fit the industrial automation domain.

The discussion presented in section 3.4 supported by the state-of-the-art in SOA-based approaches for industrial automation device level (chapter 3) showed that there are still several research and development challenges to be addressed in order to better blend both domains into an unified solution following a "best of both worlds" combination. The revised state-of-the-art already demonstrated the soundness of pushing SOA approach into the industrial automation device level, and in particular to systems integrator assistance during system lifecycle interventions.

In this context and as depicted in **C.1**, the current work presents a **discussion on how SOA aspects can be employed or adapted to tackle these issues exposing all major advantages, drawbacks and open aspects**. This study was conducted by the author while integrated in several major R&D projects on the domain of SOA in industrial automation such as SODA and SOCRADES. This contribution (**C.1**) is also the outcome of numerous interactions with key players in the domain as well as with innovative end-users that exposed a broader and critical view over the real requirements and expectations for the future to come.

RQ.2 – Which tools and services are essential to support device lifecycle assistance of a service-oriented application in industrial automation?

Taking into account **C.1**, the pursued hypothesis for this research question is:

H.2 – The ability to assist and even automate reengineering interventions along the lifecycle of a service-oriented industrial automation system is improved if a modular and customizable service-oriented reference architecture is available and composed by an interoperable set of tools and services aimed to support the device lifecycle management

in a agile way.

The reference architecture described in chapter 4 followed the above premises of modularity and customization to adapt to dissimilar system requirements and assist the systems integrator during lifecycle reengineering interventions (**C.2**). In the scope of device lifecycle support, the several strands of systems integrator assistance scope inspired each architecture element. The pursued prototype implementations showed not only its feasibility but also its **added value when addressing device level discovery, access, management and monitoring** in the scope of distinct service-oriented industrial automation demonstrators. Included in this contribution, **the introduction of semantic web concepts revealed essential to solve knowledge mismatches during reengineering interventions and execution of production processes** (**C.2.1**).

RQ.3 – What device model should be employed to better exploit SOA at device level in service-oriented industrial automation?

To support previous reference architecture (**C.2**), there was a need to create a suitable device model to enable its objectives at device level. This device model was investigated pursuing the following hypothesis:

H.3 – At device level in industrial automation, service-oriented compliancy and "plug-and-play" agility are increased if a service-oriented device model is specified comprising a set of built-in generic services to support agile discovery, identification, setup, monitoring, diagnosis along with the ability to deploy and manage application-specific services.

Chapter 5 introduced the proposed service-oriented device model. The implementations done in the scope of ITEA SODA demonstrator confirmed **the prominence of the proposed model in the scope of a innovative service-oriented industrial automation ecosystem** (**C.3**). This model **endorsed out-of-the-box interoperability and manageability mostly enhanced by the set of embedded built-in generic services**, which also included the proposed deployment service specification (**C.3.1**). This feature allowed a standard-based way of customizing devices in accordance with actual system requirements. The proposed device model also allowed the employment of more traditional control solutions due to the **encapsulation solution that clearly detaches interfaces from concrete implementations** (**C.3.2**). Taking into account that DPWS and OPC UA are currently the two most promising approaches to deploy SOA at device level, the author also assessed both specifications and specified **a merge solution envisioning an unified specification** (**C.3.3**) fully compliant with the proposed service-oriented device model. This approach is currently being pursued and extended in the scope of AESOP project [Bony et al., 2011].

To finalize, the current work embraces both a reference architecture and a device model inspired by the results of the present survey and discussion regarding the application of SOA principles, methodologies and technology to the industrial automation domain, and more specifically to the device lifecycle support. The proposed architecture includes a distributed range of modules that can be composed and seamlessly integrated to deliver a customized environment to turn reengineering and other operative interventions more agile in the scope of a service-oriented industrial application. The architecture exploits the combination of SOA principles and semantic web techniques with the industrial automation latest requirements and visions for the future. The proposed device model by relying on a set of built-in services and the ability to deploy application-specific services through a generic and open interface revealed to be the valuable enabler for more complex device management and control. The proposed solution enables an improved out-of-the-box interoperability and customization. Altogether, the present work delivers a relevant proposal to push SOA-based solutions into the scope of industrial automation device level. The work was validated by two independent prototype implementations and widely recognized either by the R&D partners from major international projects of the area in which the author was deeply involved and through its publication on reference journals and conferences.

As a short final summary, these are the principal contributions of the current work:

- **C.1 – Survey and assessment of SOA in Industrial Automation device level**
  A guide and discussion on how to SOA aspects can be employed and adapted to tackle some of the existing industrial automation issues at device level.

- **C.2 – Device Lifecycle support Architecture**
  Supported by C.1, the proposed architecture is composed by a set of service-oriented entities that can be combined to improve systems integrator assistance along application and device lifecycle in terms of discovery, access, management and monitoring.

  - **C.2.1 – Semantic Assistance**
    Integration of web semantic techniques to handle information uncertainty and mismatch during reengineering interventions and execution of production processes.

- **C.3 – Service-oriented Device Model**
  An innovative SOA-based device model to enable and extend reference architecture principles and goals.

  - **C.3.1 – Generic Services**
    Generic device functions as services to enhance device out-of-the-box interoperability as well as the ability to deploy new customised services in a open service-oriented fashion to fit current demands.

  - **C.3.2 – Domain backward compliance**
    Clear separation between service interface and concrete implementation to enable the employment of existing knowledge and programming languages by domain experts.
  - **C.3.3 – Merge of DPWS & OPC UA specifications**
    Merge proposal for an unified specification envisioning a more complete SOA middleware solution

7.3 Associated Challenges and Constraints

The current work presented a new input to enable a more agile support for industrial automation device level along system lifecycle. By laying over open web standards and focusing on interoperability, modularity, uncomplicated management and adaptation, particularly enhanced by the use of the dynamic deployment built-in service, it is possible to define a set of components that together form a customizable device level support infrastructure. Each infrastructure entity exposes its services in the network, which will enable a mutual transparent interoperability and a customized composition of modules to face each particular system requirements. The proposed device model also plays a major role on the support of the overall system evolution by embedding out-of-the-box services that enable transparent discovery, identification, setup, and management and, at the same time, allow effortless device and corresponding services customization.

Even if the current approach presented a reference architecture and a device model proposals there are still some open challenges that should be taken into account when deploying it into a concrete real world application:

- *Determining the appropriate "subset" of SOA that fits industrial automation device level:* as discussed in chapter 3, SOA lately encompasses a wide spectrum of applications, specifications and arguments from different research communities. Although along chapter 3 the author investigated the related work on the domain of industrial automation device level and provided a set of open research challenges and guidelines to increase the adoption of SOA at this level, both domains are continuously evolving. This situation implies a constant need to research the state-of-the-art to check for current proposal cogency. However, it is then important to wisely choose which sub-approaches make sense to be also pushed into the industrial automation device level to extend current proposal and what would be the implications of this future increment.

- *Adapting service-oriented modelling and granularity to device heterogeneity:* due to the considerable diversity of device level characteristics and scope of application, the modelling of a service-oriented application and according services granularity remains a subjective area. In section 3.3 the author provides some recommendations

on how to leverage coarse- and fine-grained services in the context of industrial automation device level and in particular for devices with low-resources. However, even if strict governance and versioning policies are inaugurated to clearly define design and development heuristics and specifications to follow along system lifecycle as discussed in section 2.2.4, there is always an associated risk related with the partially subjective evaluation and decision from system expert.

- *Defining the right balance between innovation and existing domain know-how:* whenever a new paradigm or technology emerges and substitutes or overlaps previous prevailing solution there is always a risk to loose some of the accumulated training and knowledge. The introduction of SOA in this domain promises to augment the level of abstraction when composing the service-oriented application and hide implementation intricacies in a bottom-up progressive manner. The big question is where to draw the line where both domains will blend – a discussion on this subject is depicted in section 3.4.

- *Combining open standards and interfaces with proprietary added-value:* even if SOA endorses the use of open web standards and interfaces, there is a comprehensible propensity for device manufacturers, in particular reference players, to develop their own standards and proprietary features to distinguish their commercial offer from their main competitors. Although this tendency may lead to technology improvements at device level, there is a risk to compromise some of the SOA principles such as openness, interoperability or reusability.

- *Open source availability:* Although the availability of open source solutions is already partly achieved, there is still some work to do in order to create a large community of developers and users. Also, major embedded OS vendors such as Microsoft or Windriver still need to evolve their own range of products to support these new industrial paradigms.

- *Unfamiliar concepts and suspicion in the domain of application:* When introducing unfamiliar concepts and technology into a new domain as conservative and resource-critical as industrial automation, the security aspects must be taken into account. The open and distributed nature of SOA applications including the one presented in this work can transmit apprehension to the system expert and reduce its willingness to accept a solution that does not assure him in terms of security of access or on avoiding the undesirable release of proprietary know-how. Although there is no technical issue behind supporting these security options using WS–* specifications, this item is still a major roadblock when pushing these ideas into the industrial automation domain.

- *Education and Training:* In this context, the education aspects also play a fundamental role to educate and train the industrial automation community about the new

trends in new research areas and show how these can be transferred into real production installations without misconceptions or uncertainties. The present work is also included in this scope and transmits a clarification on how to employ SOA concepts in this domain without losing some of the established know-how and it is expected to allow a more straightforward adoption of these concepts and technology.

7.4 Future work

The current work also led to the emergence of new challenges and subjects to be further investigated in the context of device lifecycle support in service-oriented industrial automation. These subject are mostly related with supplementary extensions and improvements to current architecture and device model as well as research challenges to be tackled in the context of SOA-based applications:

- Implementation of a unified DPWS / OPC UA stack and validation through the deployment on real-world industrial automation scenarios. The approach depicted in this work was considered a relevant starting-point for further refinement and extension.

- Investigation of security-related solutions to ensure information confidentiality and access rights, particularly when accessing and modifying devices configuration, invoking critical services or exchanging proprietary information. These solutions must take into account the available computing capabilities of low-resources devices and do not compromise overall system integrity and performance.

- Research and development of methods and tool to ease up the assembly of the device lifecycle support infrastructure itself to fit particular system requirements in a similar composition approach as a regular service-oriented application based on BPEL, BPMN or other emerging standard.

- The current proposal included the retrieval of devices topology composed in a hierarchical form only based on local information. An extension to this feature would allow the retrieval of more complex interaction patterns such as choreography-based interaction. This interaction graph could provide a deeper visualization and understanding over the core behaviour of the system as a whole.

- Ability to deploy services into devices based on other industrial automation technologies such as IEC61499 or CANopen encapsulated in *ServiceClass* instances.

- Integration and adaptation of process modelling tools to support more complex workflow logic and better visual interfaces

- Extensively explore the fitness of different granularity levels on several industrial platforms and retrieve the heuristics to automatically assist systems integrator during modeling phase.
- Evaluate the suitability, relevance and performance aspects of exploiting the proposed solution to a domain composed by a majority of wireless devices.

Bibliography

[AESOP, 2012] AESOP (2012). FP7-IMC AESOP project website. http://www.imc-aesop.eu/.

[Al-Safi and Vyatkin, 2007] Al-Safi, Y. and Vyatkin, V. (2007). An ontology-based reconfiguration agent for intelligent mechatronic systems. In *Holonic and Multi-Agent Systems for Manufacturing*, volume 4659, pages 114–126. Springer Berlin Heidelberg.

[Alves et al., 2006] Alves, A., Arkin, A., Askary, S., Bloch, B., Curbera, F., Goland, Y., Kartha, N., König, D., Mehta, V., Thatte, S., et al. (2006). Web Services Business Process Execution Language – version 2.0. Available at https://www.oasis-open.org/committees/wsbpel/.

[Alves et al., 2000] Alves, M., Tovar, E., and Vasques, F. (2000). Ethernet goes real-time: a survey on research and technological developments. Technical report, Polytechnic Institute of Porto, School of Engineering.

[ARC, 2009] ARC (2009). PcVue Solutions. http://www.arcinfo.com/.

[Arkin et al., 2002] Arkin, A., Askary, S., Fordin, S., Jekeli, W., Kawaguchi, K., Orchard, D., Pogliani, S., Riemer, K., Struble, S., Takacsi-Nagy, P., et al. (2002). Web Service Choreography Interface (WSCI) 1.0, 2002. Available at http://www.w3.org/TR/wsci.

[Arora et al., 2004] Arora, A., Cohen, J., Davis, J., Golovinsky, E., He, J., Hines, D., McCollum, R., Milenkovic, M., Montgomery, P., Schlimmer, J., et al. (2004). Web Services for Management (WS-Management) Specification. Available at http://www.dmtf.org/standards/wsman.

[Arrowhead, 2012] Arrowhead (2012). ARTEMIS – Arrowhead project website. http://www.arrowhead.eu/.

[Avilés-López and García-Macías, 2009] Avilés-López, E. and García-Macías, J. (2009). TinySOA: a service-oriented architecture for wireless sensor networks. *Service Oriented Computing and Applications*, 3(2):99–108.

[Babbie, 1990] Babbie, E. (1990). *Survey research methods*. Wadsworth Pub. Co., 2 edition.

[Bakken, 2001] Bakken, D. (2001). Middleware. *Encyclopedia of Distributed Computing*.

[Barata, 2004] Barata, J. (2004). *Coalition based approach for shop floor agility – a multi-agent approach*. PhD thesis, Universidade Nova de Lisboa.

[Barata and Camarinha-Matos, 2003] Barata, J. and Camarinha-Matos, L. (2003). Coalitions of manufacturing components for shop floor agility-the CoBASA architecture. *International Journal of Networking and Virtual Organisations*, 2(1):50–77.

[Barata et al., 2008] Barata, J., Camarinha-Matos, L., and Cândido, G. (2008). A multiagent-based control system applied to an educational shop floor. *Robotics and Computer-Integrated Manufacturing*, 24(5):597–605.

[Barata et al., 2007a] Barata, J., Onori, M., Frei, R., and Leitão, P. (2007a). Evolvable Production Systems: Enabling Research Domains. In *International Conference on Changeable, Agile, Reconfigurable, and Virtual Production*, pages 239–244, Toronto, Ontario, Canada.

[Barata et al., 2007b] Barata, J., Ribeiro, L., and Colombo, A. (2007b). Diagnosis using service oriented architectures (SOA). In *International Conference on Industrial Informatics*, volume 2, pages 1203–1208. IEEE.

[Baresi et al., 2003] Baresi, L., Heckel, R., Thöne, S., and Varró, D. (2003). Modeling and validation of Service-oriented Architectures: Application vs. Style. *ACM SIGSOFT Software Engineering Notes*, 28(5):68–77.

[Barisic et al., 2007] Barisic, D., Krogmann, M., Stromberg, G., and Schramm, P. (2007). Making embedded software development more efficient with SOA. In *International Conference on Advanced Information Networking and Applications Workshops*, volume 1, pages 941–946. IEEE.

[Barros et al., 2005] Barros, A., Dumas, M., and Oaks, P. (2005). A critical overview of the Web Services Choreography Description Language. *BPTrends Newsletter*, 3.

[Beckhoff, 2008] Beckhoff (2008). WSD: plug-and-play for building automation – Beckhoff News. Available at http://www.beckhoff.com/english.asp?press/news0408.htm.

[Bedworth et al., 1991] Bedworth, D., Henderson, M., and Wolfe, P. (1991). *Computer-integrated design and manufacturing*. McGraw-Hill, Inc.

[Beisiegel et al., 2005] Beisiegel, M., Blohm, H., Booz, D., Dubray, J., and Colyer, A. (2005). Service Component Architecture – Building Systems using Service Oriented Architecture. Whitepaper.

[Bell, 2008] Bell, M. (2008). *Service-oriented modeling: service analysis, design, and architecture*. John Wiley & Sons, Inc.

[BEMO-COFRA, 2012] BEMO-COFRA (2012). FP7-ICT BEMO-COFRA project website. http://www.bemo-cofra.eu/.

[Beran et al., 2010] Beran, J., Fiedler, P., and Zezulka, F. (2010). Virtual automation networks. *IEEE Industrial Electronics Magazine*, 4(3):20–27.

[Berners-Lee and Fischetti, 1999] Berners-Lee, T. and Fischetti, M. (1999). *Weaving the Web: The original design and ultimate destiny of the World Wide Web by its inventor*. Harper San Francisco.

[Bernstein and Haas, 2008] Bernstein, P. and Haas, L. (2008). Information integration in the enterprise. *Communications of the ACM*, 51(9):72–79.

[Bertolino, 2007] Bertolino, A. (2007). Software testing research: Achievements, challenges, dreams. In *Future of Software Engineering*, pages 85–103. IEEE.

[Bertolino et al., 2011] Bertolino, A., De Angelis, G., Sabetta, A., and Polini, A. (2011). Trends and Research Issues in SOA Validation. *Performance and Dependability in Service Computing: Concepts, Techniques and Research Directions*.

[Bi et al., 2007] Bi, Z., Lang, S., and Shen, W. (2007). Reconfigurable manufacturing systems: the state of the art. *International Journal of Production Research*, 46(4):967–992.

[Bichier and Lin, 2006] Bichier, M. and Lin, K. (2006). Service-oriented computing. *Computer*, 39(3):99–101.

[Biffl et al., 2009] Biffl, S., Schatten, A., and Zoitl, A. (2009). Integration of heterogeneous engineering environments for the automation systems lifecycle. In *International Conference on Industrial Informatics*, pages 576–581. IEEE.

[Bloomberg and Schmelzer, 2006] Bloomberg, J. and Schmelzer, R. (2006). *Service orient or be doomed*. Wiley.

[Bogdan and Biklen, 1982] Bogdan, R. and Biklen, S. (1982). *Qualitative research for education*. Allyn and Bacon Boston.

[Bony et al., 2011] Bony, B., Harnischfeger, M., and Jammes, F. (2011). Convergence of OPC UA and DPWS with a cross-domain data model. In *International Conference on Industrial Informatics*, pages 187–192. IEEE.

[Boterenbrood, 2000] Boterenbrood, H. (2000). CANopen, high-level protocol for CANbus. Technical report, NIKHEF.

[Brehmer and Wang, 1999] Brehmer, N. and Wang, C. (1999). Reconfigurable manufacturing systems and environment consciousness. In *International Symposium on Environmentally Conscious Design and Inverse Manufacturing*, pages 463–468.

[Brogi et al., 2004] Brogi, A., Canal, C., Pimentel, E., and Vallecillo, A. (2004). Formalizing web service choreographies. *Electronic Notes in Theoretical Computer Science*, 105:73–94.

[Brooks, 2001] Brooks, P. (2001). Ethernet/IP: Industrial Protocol. In *Emerging Technologies in Factory Automation*. IEEE.

[Brown and Ellis, 2004] Brown, K. and Ellis, M. (2004). Best practices for web services versioning. Available at http://www.ibm.com/developerworks/webservices/library/ws-version/.

[Brundtland, 1987] Brundtland, G. (1987). Our common future. *Oxford paperbacks*, 42(427).

[Bruneo et al., 2007] Bruneo, D., Puliafito, A., and Scarpa, M. (2007). Mobile middleware: Definition and motivations. *The Handbook of Mobile Middleware*, pages 145–167.

[Bussmann et al., 2004] Bussmann, S., Jennings, N., and Wooldridge, M. (2004). *Multiagent systems for manufacturing control: a design methodology*. Springer-Verlag New York Inc.

[Byrne et al., 2010] Byrne, J., Heavey, C., and Byrne, P. (2010). A review of web-based simulation and supporting tools. *Simulation modelling practice and theory*, 18(3):253–276.

[Cachapa et al., 2007] Cachapa, D., Colombo, A., Feike, M., and Bepperling, A. (2007). An approach for integrating real and virtual production automation devices applying the service-oriented architecture paradigm. In *Conference on Emerging Technologies and Factory Automation*, pages 309–314. IEEE.

[Cachapa et al., 2011] Cachapa, D., Harrison, R., and Colombo, A. (2011). Configuration of SoA-based devices in virtual production cells. *International Journal of Production Research*, 49(24):7397–7423.

[Caete et al., 2008] Caete, E., Chen, J., Díaz, M., Llopis, L., and Rubio, B. (2008). Useme: A service-oriented framework for wireless sensor and actor networks. In *International Workshop on Applications and Services in Wireless Networks*, pages 47–53. IEEE.

[Camarinha-Matos and Afsarmanesh, 2005] Camarinha-Matos, L. and Afsarmanesh, H. (2005). Collaborative networks: A new scientific discipline. *Journal of Intelligent Manufacturing*, 16(4):439–452.

[Cândido et al., 2011] Cândido, G., Colombo, A., Barata, J., and Jammes, F. (2011). Service-oriented infrastructure to support the deployment of evolvable production systems. *IEEE Transactions on Industrial Informatics*, 7(4):759–767.

[Cândido et al., 2010] Cândido, G., Jammes, F., Barata, J., and Colombo, A. (2010). SOA at device level in the industrial domain: Assessment of OPC UA and DPWS specifications. In *International Conference on Industrial Informatics*, pages 598–603. IEEE.

[Cannata et al., 2008] Cannata, A., Gerosa, M., and Taisch, M. (2008). A Technology Roadmap on SOA for smart embedded devices: Towards intelligent systems in manufacturing. In *International Conference on Industrial Engineering and Engineering Management*, pages 762–767. IEEE.

[Cardoso, 2007] Cardoso, J. (2007). *Semantic Web services: theory, tools, and applications*. Information Science Publishing.

[Cerami and St Laurent, 2002] Cerami, E. and St Laurent, S. (2002). *Web services essentials*. O'Reilly & Associates, Inc.

[Cerf and Kahn, 1974] Cerf, V. and Kahn, R. (1974). A Protocol for Packet Network Intercommunication. *IEEE Transactions on Communication*, 22(5).

[Chappell, 2004] Chappell, D. (2004). *Enterprise Service Bus*. O'Reilly Media, Inc.

[Chen et al., 2008] Chen, D., Doumeingts, G., and Vernadat, F. (2008). Architectures for enterprise integration and interoperability: Past, present and future. *Computers in Industry*, 59(7):647–659.

[Chen et al., 2006] Chen, X., Cai, W., Turner, S., and Wang, Y. (2006). SOAr-DSgrid: Service-oriented architecture for distributed simulation on the grid. In *Workshop on Principles of Advanced and Distributed Simulation*, pages 65–73. IEEE Computer Society.

[Chen, 2007] Chen, Y. (2007). Modeling and simulation for and in service-oriented computing paradigm. *Simulation-Transactions of the Society for Modelling and Simulation Interl*, 83(1):3–6.

[Choi et al., 2006] Choi, N., Song, I., and Han, H. (2006). A survey on ontology mapping. *ACM Sigmod Record*, 35(3):34–41.

[Christensen, 1994] Christensen, J. (1994). Holonic manufacturing systems: initial architecture and standards directions. In Consortium, E. H., editor, *Workshop on Holonic Manufacturing Systems*.

[Cohen et al., 2007] Cohen, J., Davis, D., Kreger, H., Tewari, V., and Walker, M. (2007). Management Harmonization Overview. In *OGF19 – Open Grid Forum*, North Carolina, USA.

[Colombo et al., 2005] Colombo, A., Jammes, F., Smit, H., Harrison, R., Lastra, J., and Delamer, I. (2005). Service-oriented architectures for collaborative automation. In *Annual Conference of IEEE Industrial Electronics Society*, pages 2649–2654. IEEE.

[Colombo and Karnouskos, 2009] Colombo, A. and Karnouskos, S. (2009). Towards the factory of the future: A service-oriented cross-layer infrastructure. *ICT Shaping the World: A Scientific View. European Telecommunications Standards Institute (ETSI), John Wiley and Sons*, 65:81.

[Colombo et al., 2004] Colombo, A., Schoop, R., Leitão, P., and Restivo, F. (2004). A collaborative automation approach to distributed production systems. In *International Conference on Industrial Informatics*. IEEE.

[COSMOS, 2012] COSMOS (2012). FP7-NMP COSMOS project website. http://www.cosmosproject.eu/.

[Creswell, 2009] Creswell, J. (2009). *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage Publications, Inc.

[Creswell et al., 2003] Creswell, J., Tashakkori, A., Jensen, K., and Shapley, K. (2003). Teaching mixed methods research: Practices, dilemmas, and challenges. *Handbook of mixed methods in social and behavioral research*, pages 619–637.

[Cucinotta et al., 2009] Cucinotta, T., Mancina, A., Anastasi, G., Lipari, G., Mangeruca, L., Checcozzo, R., and Rusina, F. (2009). A real-time service-oriented architecture for industrial automation. *IEEE Transactions on Industrial Informatics*, 5(3):267–277.

[Curbera et al., 2002] Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., and Weerawarana, S. (2002). Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI. *IEEE Internet Computing*, 6(2):86–93.

[Daconta et al., 2009] Daconta, M., Obrst, L., and Smith, K. (2009). The Semantic Web: a guide to the future of XML, Web services, and knowledge management. *Computing Reviews*, 45(12):778–779.

[Davies, 2007] Davies, S. (2007). The fundamentals of Ethernet/IP. *Computing and Control Engineering*, 18(1):42–45.

[De Souza et al., 2008] De Souza, L., Spiess, P., Guinard, D., Köhler, M., Karnouskos, S., and Savio, D. (2008). SOCRADES: A web service based shop floor integration infrastructure. In *International conference on The internet of things*, pages 50–67. Springer-Verlag.

[Decision - Etudes and Conseil RPA, 2008] Decision - Etudes and Conseil RPA (2008). Monitoring and Control: Today's Market, its Evolution till 2020 and the Impact of ICT on these. Technical report, European Commission DG Information Society and Media.

[Decotignie, 2005] Decotignie, J. (2005). Ethernet-based real-time and industrial communications. *Proceedings of the IEEE*, 93(6):1102–1117.

[Decotignie, 2009] Decotignie, J. (2009). The many faces of industrial Ethernet [past and present]. *IEEE Industrial Electronics Magazine*, 3(1):8–19.

[Deering and Hinden, 1998] Deering, S. and Hinden, R. (1998). Internet Protocol, Version 6 (IPv6) - Specification. DARPA Internet Program – Available at http://www.ietf.org/rfc/rfc2460.txt.

[DeFee, 2004] DeFee, J. (2004). Simulation on demand: Using SIMPROCESS in an SOA Environment. *Business Process Trends, White paper*.

[Delamer, 2006] Delamer, I. (2006). *Event-based middleware for reconfigurable manufacturing systems: a semantic web services approach*. PhD thesis, Tampere University of Technology.

[Delamer and Lastra, 2006a] Delamer, I. and Lastra, J. (2006a). Quality of service for CAMX middleware. *International Journal of Computer Integrated Manufacturing*, 19(8):784–804.

[Delamer and Lastra, 2006b] Delamer, I. and Lastra, J. (2006b). Self-orchestration and choreography: Towards architecture-agnostic manufacturing systems. In *International Conference on Advanced Information Networking and Applications*, volume 2, pages 573–582. IEEE Computer Society.

[Delamer and Lastra, 2007] Delamer, I. and Lastra, J. (2007). Loosely-coupled automation systems using device-level SOA. In *International Conference on Industrial Informatics*, volume 2, pages 743–748. IEEE.

[Delsing et al., 2011] Delsing, J., Eliasson, J., Kyusakov, R., Colombo, A. W., Jammes, F., Nessaether, J., Karnouskos, S., and Diedrich, C. (2011). A migration approach towards a soa-based next generation process control and monitoring. In *Annual Conference on IEEE Industrial Electronics Society*, pages 4472–4477. IEEE.

[Desai et al., 2006] Desai, N., Mallya, A., Chopra, A., and Singh, M. (2006). OWL-P: a methodology for business process development. In *International Bi-Conference Workshop – Agent-Oriented Information Systems III*, pages 79–94. Springer Berlin Heidelberg.

[Di Nitto et al., 2008] Di Nitto, E., Ghezzi, C., Metzger, A., Papazoglou, M., and Pohl, K. (2008). A journey to highly dynamic, self-adaptive service-based applications. *Automated Software Engineering*, 15(3):313–341.

[Dibowski and Kabitzsch, 2008] Dibowski, H. and Kabitzsch, K. (2008). Semantic device descriptions based on standard semantic web technologies. In *International Workshop on Factory Communication Systems*, pages 395–404. IEEE.

[Dimitrov et al., 2007] Dimitrov, M., Simov, A., Momtchev, V., and Konstantinov, M. (2007). WSMO Studio – A Semantic Web Services Modelling Environment for WSMO. *The Semantic Web: Research and Applications*, pages 749–758.

[DMTF, 2003] DMTF (2003). CIM Schema. http://www.dmtf.org/standards/cim/.

[Dustdar and Schreiner, 2005] Dustdar, S. and Schreiner, W. (2005). A survey on web services composition. *International Journal of Web and Grid Services*, 1(1):1–30.

[Egger and Carpi, 2008] Egger, A. E. and Carpi, A. (2008). Research methods: The practice of science. Visionlearning.

[ElMaraghy, 2005] ElMaraghy, H. (2005). Flexible and reconfigurable manufacturing systems paradigms. *International journal of flexible manufacturing systems*, 17(4):261–276.

[Enns, 2006] Enns, R. (2006). NETCONF Configuration Protocol. Available at http://tools.ietf.org/html/rfc4741.

[EPES, 2012] EPES (2012). FP7-ICT EPES project website. http://www.epes-project.eu/.

[Erl, 2005] Erl, T. (2005). *Service-oriented architecture: concepts, technology, and design*. Prentice Hall PTR Upper Saddle River, NJ, USA.

[Erl, 2007] Erl, T. (2007). *SOA: Principles of service design*. Prentice Hall Press Upper Saddle River, NJ, USA.

[eSonia, 2012] eSonia (2012). ARTEMIS - eSonia project website. http://www.esonia.eu/.

[Estrem, 2003] Estrem, W. (2003). An evaluation framework for deploying web services in the next generation manufacturing enterprise. *Robotics and Computer-Integrated Manufacturing*, 19(6):509–519.

[Eurostat, 2012] Eurostat (2012). Euro Indicators – Industrial Production (December 2011). News Release 24/2012, Eurostat Press Office.

[Evdemon, 2005] Evdemon, J. (2005). Principles of service design: Service versioning. Microsoft Corporation – Available at http://msdn.microsoft.com/en-us/library/ms954726.aspx.

[ExtremeFactories, 2012] ExtremeFactories (2012). FP7-ICT ExtremeFactories project website. http://www.extremefactories.eu/.

[Fang et al., 2007] Fang, R., Lam, L., Fong, L., Frank, D., Vignola, C., Chen, Y., and Du, N. (2007). A version-aware approach for web service directory. In *International Conference on Web Services*, pages 406–413. IEEE.

[Feenstra and Hanson, 1996] Feenstra, R. and Hanson, G. (1996). Globalization, outsourcing, and wage inequality. *The American Economic Review*, 86(2):240–245.

[Feit, 1993] Feit, S. (1993). *SNMP: A guide to network management*. McGraw-Hill Professional.

[Feldhorst et al., 2009] Feldhorst, S., Libert, S., ten Hompel, M., and Krumm, H. (2009). Integration of a Legacy Automation System into a SOA for Devices. In *Conference on Emerging Technologies & Factory Automation*, pages 1–8. IEEE.

[Fensel and Bussler, 2002] Fensel, D. and Bussler, C. (2002). Semantic web enabled web services. In *Advances in artificial intelligence: 25th Annual German Conference on AI*, volume 2479, page 316. Springer Verlag.

[Fensel et al., 2011a] Fensel, D., Facca, F., Simperl, E., and Toma, I. (2011a). Semantic web. *Semantic Web Services*, pages 87–104.

[Fensel et al., 2011b] Fensel, D., Facca, F., Simperl, E., and Toma, I. (2011b). *Semantic web services*. Springer-Verlag New York Inc.

[Fielding and Taylor, 2002] Fielding, R. and Taylor, R. (2002). Principled design of the modern web architecture. *ACM Transactions on Internet Technology (TOIT)*, 2(2):115–150.

[FIPA, 2001] FIPA (2001). FIPA Device Ontology Specification (PC00091A). Available at http://www.fipa.org/specs/fipa00091/PC00091A.html.

[Flahive et al., 2011] Flahive, A., Taniar, D., and Rahayu, W. (2011). Ontology as a Service (OaaS): a case for sub-ontology merging on the cloud. *The Journal of Supercomputing*, pages 1–32.

[Foster, 2005] Foster, I. (2005). Service-oriented science. *Science*, 308(5723):814–817.

[Frei et al., 2007] Frei, R., Barata, J., and Onori, M. (2007). Evolvable Production Systems: Context and Implications. In *International Symposium on Industrial Electronics*, pages 3233–3238. IEEE.

[Geensys, 2009] Geensys (2009). Geensys ControlBuild Automation Software Platform. Info available at http://geensys.com/control-build/.

[Gennari et al., 2003] Gennari, J., Musen, M., Fergerson, R., Grosso, W., Crubézy, M., Eriksson, H., Noy, N., and Tu, S. (2003). The evolution of Protégé: an environment for knowledge-based systems development. *International Journal of Human-Computer Studies*, 58(1):89–123.

[Gilart-Iglesias et al., 2006a] Gilart-Iglesias, V., Maciá-Pérez, F., Capella-D'alton, A., and Gil-Martínez-Abarca, J. (2006a). Industrial machines as a service: A model based on

embedded devices and web services. In *International Conference on Industrial Informatics*, pages 630–635. IEEE.

[Gilart-Iglesias et al., 2006b] Gilart-Iglesias, V., Maciá-Pérez, F., Mora-Gimeno, F., and Berná-Martínez, J. (2006b). Normalization of industrial machinery with embedded devices and SOA. In *Conference on Emerging Technologies and Factory Automation*, pages 173–180. IEEE.

[Goddard, 2006] Goddard, T. (2006). Using NETCONF over the Simple Object Access Protocol (SOAP). Available at http://tools.ietf.org/html/rfc4743.

[Goldman et al., 1995] Goldman, S., Nagel, R., and Preiss, K. (1995). *Agile competitors and virtual organizations: strategies for enriching the customer*. Van Nostrand Reinhold Company.

[Gómez-Pérez et al., 2002] Gómez-Pérez, A., Corcho, O., and Fernández-López, M. (2002). *Ontological Engineering*. Springer-Verlag, London, Berlin.

[Gorbach and Nick, 2002] Gorbach, G. and Nick, R. (2002). Collaborative manufacturing management strategies. *white paper, ARC Advisory Group*.

[Gorton and Liu, 2004] Gorton, I. and Liu, A. (2004). Architectures and technologies for enterprise application integration. In *International Conference on Software Engineering*, pages 726–727. IEEE.

[Gould, 1997] Gould, P. (1997). What is agility? *Manufacturing Engineer*, 76(1):28–31.

[Greenwood, 2005] Greenwood, D. (2005). JADE Web Service Integration Gateway (WSIG). *Whitestein Technologies, Jade Tutorial, AAMAS*.

[Greenwood and Calisti, 2004] Greenwood, D. and Calisti, M. (2004). Engineering web service-agent integration. In *International Conference on Systems, Man and Cybernetics*, volume 2, pages 1918–1925. IEEE.

[Greenwood et al., 2007] Greenwood, D., Lyell, M., Mallya, A., and Suguri, H. (2007). The IEEE FIPA approach to integrating software agents and web services. In *International joint conference on Autonomous agents and multiagent systems*, page 276. ACM.

[Grossmann et al., 2008] Grossmann, D., Bender, K., and Danzer, B. (2008). OPC UA based field device integration. In *SICE Annual Conference, 2008*, pages 933–938. IEEE.

[Gruber, 1993] Gruber, T. (1993). What is an ontology? *Knowledge Acquisition*, 5(2):199–220.

[GTAI, 2012] GTAI (2012). Industry 4.0 - Germany takes first steps. Available at http://www.bmbf.de/de/19955.php.

[Gunasekaran, 1998] Gunasekaran, A. (1998). Agile manufacturing: enablers and an implementation framework. *International Journal of Production Research*, 36(5):1223–1247.

[Gunasekaran et al., 2008] Gunasekaran, A., Lai, K., and Edwin Cheng, T. (2008). Responsive supply chain: a competitive strategy in a networked economy. *Omega*, 36(4):549–564.

[Gungor and Hancke, 2009] Gungor, V. C. and Hancke, G. P. (2009). Industrial wireless sensor networks: Challenges, design principles, and technical approaches. *IEEE Transactions on Industrial Electronics*, 56(10):4258–4265.

[Guo et al., 2011] Guo, Y., Pan, Z., and Heflin, J. (2011). LUBM: A benchmark for OWL knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(2-3).

[Gupta and Goyal, 1989] Gupta, Y. and Goyal, S. (1989). Flexibility of manufacturing systems: concepts and measurements. *European Journal of Operational Research*, 43(2):119–135.

[Haarslev and Müller, 2001] Haarslev, V. and Müller, R. (2001). RACER system description. *Automated Reasoning*, pages 701–705.

[Haas and Brown, 2004] Haas, H. and Brown, A. (2004). Web services glossary. Available at www.w3.org/TR/ws-gloss/.

[Hadlich, 2006] Hadlich, T. (2006). Providing device integration with OPC UA. In *International Conference on Industrial Informatics*, pages 263–268. IEEE.

[Haesen et al., 2008] Haesen, R., Snoeck, M., Lemahieu, W., and Poelmans, S. (2008). On the definition of service granularity and its architectural impact. In *Advanced Information Systems Engineering*, pages 375–389. Springer.

[Haggerty and Seetharaman, 1998] Haggerty, P. and Seetharaman, K. (1998). The benefits of CORBA-based network management. *Communications of the ACM*, 41(10):73–79.

[Halevy et al., 2005] Halevy, A., Ashish, N., Bitton, D., Carey, M., Draper, D., Pollock, J., Rosenthal, A., and Sikka, V. (2005). Enterprise information integration: successes, challenges and controversies. In *SIGMOD International conference on Management of data*, pages 778–787. ACM.

[Haller et al., 2005] Haller, A., Cimpian, E., Mocan, A., Oren, E., and Bussler, C. (2005). WSMX – a semantic service-oriented architecture. In *International Conference on Web Services*, pages 321–328. IEEE.

[Hannelius et al., 2008] Hannelius, T., Salmenpera, M., and Kuikka, S. (2008). Roadmap to adopting OPC UA. In *International Conference on Industrial Informatics*, pages 756–761. IEEE.

[Happel and Seedorf, 2006] Happel, H. and Seedorf, S. (2006). Applications of ontologies in software engineering. In *Workshop on Sematic Web Enabled Software Engineering in ISWC*, pages 5–9.

[Harrington, 1974] Harrington, J. (1974). *Computer integrated manufacturing*. Industrial Press New York.

[Harrison and Colombo, 2005] Harrison, R. and Colombo, A. (2005). Collaborative automation from rigid coupling towards dynamic reconfigurable production systems. In *World Congress*, volume 16, pages 1570–1570.

[Hayes and Pisano, 1994] Hayes, R. and Pisano, G. (1994). The new manufacturing strategy in. *Harvard Business Review*, 72(1):77–86.

[He, 2003] He, H. (2003). What is service-oriented architecture? Available at http://www.xml.com/pub/a/ws/2003/09/30/soa.html.

[Heiler, 1995] Heiler, S. (1995). Semantic interoperability. *ACM Computing Surveys (CSUR)*, 27(2):271–273.

[Hendler, 2001] Hendler, J. (2001). Agents and the semantic web. *IEEE Intelligent Systems*, 16(2):30–37.

[Herrera et al., 2008] Herrera, V., Bepperling, A., Lobov, A., Smit, H., Colombo, A., and Lastra, J. (2008). Integration of multi-agent systems and service-oriented architecture for industrial automation. In *International Conference on Industrial Informatics*, pages 768–773. IEEE.

[Herzum and Sims, 2000] Herzum, P. and Sims, O. (2000). *Business Components Factory: A Comprehensive Overview of Component-Based Development for the Enterprise*. John Wiley & Sons, Inc.

[Huhns, 2002] Huhns, M. (2002). Agents as web services. *IEEE Internet Computing*, 6(4):93–95.

[Huhns and Singh, 2005] Huhns, M. and Singh, M. (2005). Service-oriented computing: Key concepts and principles. *IEEE Internet Computing*, 9(1):75–81.

[Hui and Culler, 2008a] Hui, J. W. and Culler, D. E. (2008a). Extending IP to low-power, wireless personal area networks. *IEEE Internet Computing*, 12(4):37–45.

[Hui and Culler, 2008b] Hui, J. W. and Culler, D. E. (2008b). IP is dead, long live IP for wireless sensor networks. In *Conference on Embedded network sensor systems*, pages 15–28. ACM.

[Huitema, 1996] Huitema, C. (1996). *IPv6: the new Internet protocol*. Prentice Hall PTR.

[IBM, 2006] IBM (2006). WSDM Management white paper. Available at http://www.ibm.com/developerworks/library/specification/ws-wsdmmgmt/.

[IBM, 2008] IBM (2008). SBLIM – Standards Based Linux Instrumentation for Manageability. Available at http://sourceforge.net/projects/sblim/.

[IDC, 2008] IDC (2008). Worldwide data integration and access software 2008-2012 forecast. Technical report, IDC.

[IEEE, 1990] IEEE (1990). Standard Glossary of Software Engineering Terminology. IEEE Computer Society – Software Engineering Technical Committee.

[IEEE, 2011] IEEE (2011). IEEE Standard 802.3 – Ethernet. Available at http://standards.ieee.org/getieee802/.

[IETF, 2012] IETF (2012). Internet Engineering Task Force (IETF). http://www.ietf.org/.

[Information Sciences Institute, 1981] Information Sciences Institute, U. o. S. C. (1981). Internet Protocol - Protocol Specification. DARPA Internet Program. Available at http://www.ietf.org/rfc/rfc791.txt.

[InLife, 2008] InLife (2008). FP7-NMP InLife project website. http://www.uninova.pt/inlife/.

[innoQ, 2007] innoQ (2007). Web Services Standards as of Q1 2007. Available at http://www.innoq.com/resources/ws-standards-poster/.

[Intel, 2009] Intel (2009). Openwsman – WS-Management for all. Available at http://www.openwsman.org/.

[IoT@Work, 2012] IoT@Work (2012). FP7 - ICT IoT@Work project website. https://www.iot-at-work.eu/.

[iProd, 2012] iProd (2012). FP7-ICT iProd project website. http://www.iprod-project.eu/.

[Izaguirre et al., 2011] Izaguirre, M., Lobov, A., and Lastra, J. (2011). OPC-UA and DPWS interoperability for factory floor monitoring using complex event processing. In *International Conference on Industrial Informatics*, pages 205–211. IEEE.

[Jammes, 2011] Jammes, F. (2011). Real time device level service-oriented architectures. In *International Symposium on Industrial Electronics*, pages 1722–1726. IEEE.

[Jammes et al., 2012] Jammes, F., Bony, B., Nappey, P., Colombo, A., Delsing, J., Eliasson, J., Kyusakov, R., Karnouskos, S., Stluka, P., and Tilly, M. (2012). Technologies for SOA-based Distributed Large Scale Process Monitoring and Control Systems. In *Annual Conference of the IEEE Industrial Electronics Society*, pages 5799–5804. IEEE.

[Jammes et al., 2005a] Jammes, F., Mensch, A., and Smit, H. (2005a). Service-oriented device communications using the Devices Profile for Web Services. In *Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing*, page 8. ACM.

[Jammes and Smit, 2005a] Jammes, F. and Smit, H. (2005a). Service-oriented architectures for devices – the SIRENA view. In *International Conference on Industrial Informatics*, pages 140–147. IEEE.

[Jammes and Smit, 2005b] Jammes, F. and Smit, H. (2005b). Service-oriented paradigms in industrial automation. *IEEE Transactions on Industrial Informatics*, 1(1):62–70.

[Jammes et al., 2005b] Jammes, F., Smit, H., Lastra, J., and Delamer, I. (2005b). Orchestration of service-oriented manufacturing processes. In *Conference on Emerging Technologies and Factory Automation*, volume 1, pages 624–632. IEEE.

[Josuttis, 2007] Josuttis, N. (2007). *SOA in practice: the art of distributed system design*. O'Reilly Media, Inc.

[Juric and Sasa, 2010] Juric, M. and Sasa, A. (2010). Version Management of BPEL Processes in SOA. In *World Congress on Services (SERVICES-1)*, pages 146–147. IEEE.

[Juric et al., 2009] Juric, M., Sasa, A., Brumen, B., and Rozman, I. (2009). WSDL and UDDI extensions for version support in web services. *Journal of Systems and Software*, 82(8):1326–1343.

[Kalogeras et al., 2006] Kalogeras, A., Gialelis, J., Alexakos, C., Georgoudakis, M., and Koubias, S. (2006). Vertical integration of enterprise industrial systems utilizing web services. *IEEE Transactions on Industrial Informatics*, 2(2):120–128.

[Karnouskos et al., 2007] Karnouskos, S., Baecker, O., De Souza, L., and Spiess, P. (2007). Integration of SOA-ready networked embedded devices in enterprise systems via a cross-layered web service infrastructure. In *Conference on Emerging Technologies and Factory Automation*, pages 293–300. IEEE.

[Karnouskos and Tariq, 2008] Karnouskos, S. and Tariq, M. (2008). An agent-based simulation of SOA-ready devices. In *International Conference on Computer Modeling and Simulation*, pages 330–335. IEEE.

[King et al., 2006] King, J., Bose, R., Yang, H., Pickles, S., and Helal, A. (2006). Atlas: A service-oriented sensor platform: Hardware and middleware to enable programmable pervasive spaces. In *Conference on local computer networks*, pages 630–638. IEEE.

[Kirkham et al., 2008] Kirkham, T., Savio, D., Smit, H., Harrison, R., Monfared, R., and Phaithoonbuathong, P. (2008). SOA middleware and automation: Services, applications and architectures. In *International Conference on Industrial Informatics*, pages 1419–1424. IEEE.

[Koestler, 1976] Koestler, A. (1976). *The ghost in the machine*. Hutchinson & Co Ltd.

[Kopecký et al., 2007] Kopecký, J., Vitvar, T., Bournez, C., and Farrell, J. (2007). SAWSDL: Semantic annotations for WSDL and XML schema. *IEEE Internet Computing*, pages 60–67.

[Koren et al., 1999] Koren, Y., Heisel, U., Jovane, F., Moriwaki, T., Pritschow, G., Ulsoy, G., and Van Brussel, H. (1999). Reconfigurable manufacturing systems. *CIRP Annals-Manufacturing Technology*, 48(2):527–540.

[Krafzig et al., 2005] Krafzig, D., Banke, K., and Slama, D. (2005). *Enterprise SOA: service-oriented architecture best practices*. Prentice Hall PTR.

[Krakowiak, 2003] Krakowiak, S. (2003). What is middleware? Available at http://middleware.objectweb.org/.

[Kulvatunyou et al., 2005] Kulvatunyou, B., Cho, H., and Son, Y. (2005). A semantic web service framework to support intelligent distributed manufacturing. *International Journal of Knowledge-based and Intelligent Engineering Systems*, 9(2):107–127.

[Kushwaha et al., 2007] Kushwaha, M., Amundson, I., Koutsoukos, X., Neema, S., and Sztipanovits, J. (2007). Oasis: A programming framework for service-oriented sensor networks. In *International Conference on Communication Systems Software and Middleware*, pages 1–8. IEEE.

[Kyusakov et al., 2011] Kyusakov, R., Mäkitaavola, H., Delsing, J., and Eliasson, J. (2011). Efficient XML Interchange in Factory Automation Systems. In *Conference on IEEE Industrial Electronics Society*, pages 4478–4483. IEEE.

[Lankhorst, 2009] Lankhorst, M. (2009). *Enterprise architecture at work: Modelling, communication and analysis*. Springer-Verlag New York Inc.

[Lara et al., 2004] Lara, R., Roman, D., Polleres, A., and Fensel, D. (2004). A conceptual comparison of WSMO and OWL-S. *Web Services*, pages 254–269.

[Lassila, 2005] Lassila, O. (2005). Using the semantic web in mobile and ubiquitous computing. *Industrial Applications of Semantic Web*, pages 19–25.

[Lastra and Delamer, 2006] Lastra, J. and Delamer, M. (2006). Semantic web services in factory automation: fundamental insights and research roadmap. *IEEE Transactions on Industrial Informatics*, 2(1):1–11.

[Laukkanen and Helin, 2003] Laukkanen, M. and Helin, H. (2003). Composing workflows of semantic web services. In *AAMAS Workshop on Web Services and Agent-Based Engineering*. Springer.

[Leavitt, 2004] Leavitt, N. (2004). Are web services finally ready to deliver? *Computer*, 37(11):14–18.

[Leiner et al., 2009] Leiner, B. M., Cerf, V. G., Clark, D. D., Kahn, R. E., Kleinrock, L., Lynch, D. C., Postel, J., Roberts, L. G., and Wolff, S. (2009). A brief history of the internet. *ACM SIGCOMM Computer Communication Review*, 39(5):22–31.

[Leitão, 2009] Leitão, P. (2009). Agent-based distributed manufacturing control: A state-of-the-art survey. *Engineering Applications of Artificial Intelligence*, 22(7):979–991.

[Leitão and Restivo, 2006] Leitão, P. and Restivo, F. (2006). ADACOR: a holonic architecture for agile and adaptive manufacturing control. *Computers in Industry*, 57(2):121–130.

[Leitner et al., 2008] Leitner, P., Michlmayr, A., Rosenberg, F., and Dustdar, S. (2008). End-to-end versioning support for web services. In *International Conference on Services Computing*, volume 1, pages 59–66. IEEE.

[Leitner and Mahnke, 2006] Leitner, S. and Mahnke, W. (2006). OPC UA – Service-oriented Architecture for Industrial Applications. *ABB Corporate Research Center. Ladenburg, Germany*.

[Lemaignan et al., 2006] Lemaignan, S., Siadat, A., Dantan, J., and Semenenko, A. (2006). MASON: A proposal for an ontology of manufacturing domain. In *Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications*, pages 195–200. IEEE.

[Leppinen et al., 1997] Leppinen, M., Pulkkinen, P., and Rautiainen, A. (1997). Java- and CORBA-based network management. *Computer*, 30(6):83–87.

[Liao et al., 2004] Liao, B., Gao, J., Hu, J., and Chen, J. (2004). A federated multi-agent system: autonomic control of web services. In *International Conference on Machine Learning and Cybernetics*, volume 1, pages 1–6. IEEE.

[Lin et al., 2006] Lin, C., Chiu, H., and Chu, P. (2006). Agility index in the supply chain. *International Journal of Production Economics*, 100(2):285–299.

[Lin and Harding, 2007] Lin, H. and Harding, J. (2007). A manufacturing system engineering ontology model on the semantic web for inter-enterprise collaboration. *Computers in Industry*, 58(5):428–437.

[Lin et al., 2004] Lin, H., Harding, J., and Shahbaz, M. (2004). Manufacturing system engineering ontology for semantic interoperability across extended project teams. *International journal of production research*, 42(24):5099–5118.

[Lin and Panahi, 2010] Lin, K. and Panahi, M. (2010). A real-time service-oriented framework to support sustainable cyber-physical systems. In *International Conference on Industrial Informatics*, pages 15–21. IEEE.

[Lindberg et al., 2007] Lindberg, B., Onori, M., and Semere, D. (2007). Evolvable Production Systems: A Position Paper. In *Swedish Production Symposium*, Gothenburg, Sweden.

[Lobov et al., 2008] Lobov, A., Puttonen, J., Herrera, V., Andiappan, R., and Lastra, J. (2008). Service oriented architecture in developing of loosely-coupled manufacturing systems. In *International Conference on Industrial Informatics*, pages 791–796. IEEE.

[Lohse et al., 2005] Lohse, N., Hirani, H., and Ratchev, S. (2005). Equipment ontology for modular reconfigurable assembly systems. *International Journal of Flexible Manufacturing Systems*, 17(4):301–314.

[Loskyll et al., 2011] Loskyll, M., Schlick, J., Hodek, S., Ollinger, L., Gerber, T., and Pirvu, B. (2011). Semantic service discovery and orchestration for manufacturing processes. In *Conference on Emerging Technologies & Factory Automation*, pages 1–8. IEEE.

[Lublinsky, 2007] Lublinsky, B. (2007). Versioning in SOA. Microsoft Architect Journal – Available at http://msdn.microsoft.com/en-us/library/bb491124.aspx.

[Luder et al., 2004] Luder, A., Peschke, J., Sauter, T., Deter, S., and Diep, D. (2004). Distributed intelligence for plant automation based on multi-agent systems: the PABADIS approach. *Production Planning and Control*, 15(2):201–212.

[Luo et al., 2005] Luo, M., Goldshlager, B., and Zhang, L. (2005). Designing and implementing Enterprise Service Bus (ESB) and SOA solutions. In *International Conference on Services Computing*, volume 2, pages xiv–vol. IEEE.

[Lyell et al., 2003] Lyell, M., Rosen, L., Casagni-Simkins, M., and Norris, D. (2003). On software agents and web services: Usage and design concepts and issues. In *International Workshop on Web Services and Agent Based Engineering*.

[Maamar et al., 2003] Maamar, Z., Sheng, Q., and Benatallah, B. (2003). Interleaving web services composition and execution using software agents and delegation. In *International Workshop on Web Services and Agent-based Engineering*.

[Mackenzie and Knipe, 2006] Mackenzie, N. and Knipe, S. (2006). Research dilemmas: Paradigms, methods and methodology. *Issues In Educational Research*, 16(2):193–205.

[Manchester University, 2012] Manchester University (2012). The OWL API. Available at http://owlapi.sourceforge.net/.

[Martin et al., 2004] Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., et al. (2004). OWL-S: Semantic markup for web services. *W3C Member Submission*.

[Martin et al., 2007] Martin, D., Paolucci, M., and Wagner, M. (2007). Towards semantic annotations of web services: OWL-S from the SAWSDL perspective. In *OWL-S Experiences and Future Developments Workshop at ESWC*.

[Mathes et al., 2009a] Mathes, M., Gartner, J., Dohmann, H., and Freisleben, B. (2009a). SOAP4IPC: a real-time SOAP engine for industrial automation. In *Euromicro International Conference on Parallel, Distributed and Network-based Processing*, pages 220–226. IEEE.

[Mathes et al., 2008] Mathes, M., Heinzl, S., and Freisleben, B. (2008). Towards a time-constrained web service infrastructure for industrial automation. In *International Conference on Emerging Technologies and Factory Automation*, pages 846–853. IEEE.

[Mathes et al., 2009b] Mathes, M., Stoidner, C., Heinzl, S., and Freisleben, B. (2009b). SOAP4PLC: web services for programmable logic controllers. In *Euromicro International Conference on Parallel, Distributed and Network-based Processing*, pages 210–219. IEEE.

[McGuinness et al., 2000] McGuinness, D., Fikes, R., Rice, J., and Wilder, S. (2000). The chimaera ontology environment. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1123–1124. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

[McGuinness et al., 2004] McGuinness, D., Van Harmelen, F., et al. (2004). OWL web ontology language overview. *W3C recommendation*, 10:2004–03.

[McIlraith et al., 2001] McIlraith, S., Son, T., and Zeng, H. (2001). Semantic web services. *IEEE Intelligent Systems*, 16(2):46–53.

[Mehrabi et al., 2000] Mehrabi, M., Ulsoy, A., and Koren, Y. (2000). Reconfigurable manufacturing systems: key to future manufacturing. *Journal of Intelligent Manufacturing*, 11(4):403–419.

[Menasce, 2002] Menasce, D. (2002). QoS issues in Web services. *Internet Computing*, 6(6):72–75.

[Mendes et al., 2009a] Mendes, J., Bepperling, A., Pinto, J., Leitão, P., Restivo, F., and Colombo, A. (2009a). Software methodologies for the engineering of Service-oriented industrial automation: the Continuum project. In *Annual IEEE International Computer Software and Applications Conference*, volume 1, pages 452–459. IEEE.

[Mendes et al., 2012] Mendes, J., Leitão, P., Colombo, A., and Restivo, F. (2012). High-level Petri nets for the process description and control in service-oriented manufacturing systems. *International Journal of Production Research*, 50(6).

[Mendes et al., 2009b] Mendes, J., Leitão, P., Restivo, F., and Colombo, A. (2009b). Service-oriented agents for collaborative industrial automation and production systems. *Holonic and Multi-Agent Systems for Manufacturing*, pages 13–24.

[Mendes et al., 2007] Mendes, J., Leitão, P., Restivo, F., Colombo, A., and Bepperling, A. (2007). Engineering of service-oriented automation systems: a survey. In *I* PROMS Conference on Innovative Production Machines and Systems*.

[Menge, 2007] Menge, F. (2007). Enterprise service bus. In *Free and open source software conference*, pages 1–6.

[Mensch and Jammes, 2008] Mensch, A. and Jammes, F. (2008). Deliverable D3.3 - Devices Profile for OPC-UA. Technical report, SOCRADES Consortium.

[Metcalfe and Boggs, 1976] Metcalfe, R. and Boggs, D. (1976). Ethernet: Distributed packet switching for local computer networks. *Communications of the ACM*, 19(7):395–404.

[Microsoft, 2009] Microsoft (2009). Windows Management Instrumentation (WMI) Architecture. Available at http://msdn.microsoft.com/en-us/library/aa394553(VS.85).aspx.

[Milagaia, 2009] Milagaia, R. (2009). DPWS middleware to support agent-based manufacturing control and simulation. Master's thesis, FCT-UNL.

[Mili et al., 2001] Mili, H., Mili, A., Yacoub, S., and Addy, E. (2001). *Reuse-Based Software Engineering-techniques*. Wiley-Interscience.

[Mönch and Stehli, 2003] Mönch, L. and Stehli, M. (2003). An ontology for production control of semiconductor manufacturing processes. *Multiagent System Technologies*, pages 1097–1097.

[Monostori et al., 2006] Monostori, L., Váncza, J., and Kumara, S. (2006). Agent-based systems for manufacturing. *CIRP Annals-Manufacturing Technology*, 55(2):697–720.

[Moritz et al., 2012] Moritz, G., Golatowski, F., Timmermann, D., and Lerche, C. (2012). Beyond 6LoWPAN: Web Services in Wireless Sensor Networks. *IEEE Transactions on Industrial Informatics*.

[Moritz et al., 2008] Moritz, G., Pruter, S., Timmerman, D., and Golatowski, F. (2008). Real-time service-oriented communication protocols on resource constrained devices. In *International Multiconference on Computer Science and Information Technology*, pages 695–701. IEEE.

[Moritz et al., 2010] Moritz, G., Timmermann, D., Stoll, R., and Golatowski, F. (2010). Encoding and Compression for the Devices Profile for Web Services. In *International Workshop on Service Oriented Architectures in Converging Networked Environment*. IEEE.

[Moyne and Tilbury, 2007] Moyne, J. and Tilbury, D. (2007). The emergence of industrial control networks for manufacturing control, diagnostics, and safety data. *Proceedings of the IEEE*, 95(1):29–47.

[MSEE, 2012] MSEE (2012). FP7-ICT MSEE project website. http://www.msee-ip.eu/.

[Mulligan, 2007] Mulligan, G. (2007). The 6LoWPAN architecture. In *Workshop on Embedded networked sensors*, pages 78–82. ACM.

[Mulligan and Gracanin, 2009] Mulligan, G. and Gracanin, D. (2009). A comparison of SOAP and REST implementations of a service based interaction independence middleware framework. In *Winter Simulation Conference (WSC)*, pages 1423–1432. IEEE.

[NEMOCODED, 2012] NEMOCODED (2012). ITEA2 NEMO&CODED project website. http://www.itea2.org/project/index/view/?project=1131.

[Neumann, 2007] Neumann, P. (2007). Communication in industrial automation – what is going on? *Control Engineering Practice*, 15(11):1332–1347.

[Nguyen and Kowalczyk, 2007] Nguyen, X. and Kowalczyk, R. (2007). Ws2jade: Integrating web service with jade agents. *Service-Oriented Computing: Agents, Semantics, and Engineering*, pages 147–159.

[Noble, 2011] Noble, D. (2011). *Forces of production: A social history of industrial automation*. Transaction Publishers.

[Novakouski et al., 2012] Novakouski, M., Lewis, G., Anderson, W., and Davenport, J. (2012). Best practices for artifact versioning in service-oriented systems. Technical report, Software Engineering Institute – Carnegie Mellon.

[Noy and Musen, 2003] Noy, N. and Musen, M. (2003). The PROMPT suite: interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies*, 59(6):983–1024.

[Noy et al., 2000] Noy, N., Musen, M., et al. (2000). Algorithm and tool for automated ontology merging and alignment. Technical report, SMI.

[Nylund and Andersson, 2010] Nylund, H. and Andersson, P. (2010). Simulation of service-oriented and distributed manufacturing systems. *Robotics and Computer-Integrated Manufacturing*, 26(6):622–628.

[OASIS, 2006a] OASIS (2006a). OASIS Web Services Security (WSS). Available at https://www.oasis-open.org/committees/wss/.

[OASIS, 2006b] OASIS (2006b). Web Services Distributed Management (WSDM). Available at https://www.oasis-open.org/committees/wsdm/. OASIS WSDM Technical Committee.

[OASIS, 2007] OASIS (2007). Web Services Business Process Execution Language. Available at https://www.oasis-open.org/committees/wsbpel/.

[OASIS, 2009a] OASIS (2009a). Devices Profile for Web Services Version 1.1 Specification. Available at http://www.oasis-open.org/committees/ws-dd.

[OASIS, 2009b] OASIS (2009b). OASIS Web Services Dynamic Discovery (WS-Discovery) Version 1.1. http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01.

[OASIS, 2010] OASIS (2010). Semantic Execution Environment. Available at https://www.oasis-open.org/committees/semantic-ex.

[OASIS, 2012a] OASIS (2012a). Advancing Open Standards for the Information Society (OASIS). https://www.oasis-open.org/.

[OASIS, 2012b] OASIS (2012b). Web Services Interoperability organization (WS-I). http://www.ws-i.org/.

[Oberkampf and Roy, 2010] Oberkampf, W. and Roy, C. (2010). *Verification and Validation in Scientific Computing*. Cambridge University Press.

[Okino, 1993] Okino, N. (1993). Bionic manufacturing systems. In Peklenik, J., editor, *Conference on Flexible Manufacturing Systems, Past, Present-Future*.

[Omair Shafiq et al., 2005] Omair Shafiq, M., Ali, A., Farooq Ahmad, H., and Suguri, H. (2005). Agentweb gateway – a middleware for dynamic integration of multi agent system and web services framework. In *International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise*, pages 267–268. IEEE.

[Onori, 2002] Onori, M. (2002). Evolvable Assembly Systems – A New Paradigm? In *International Symposium on Robotics (ISR2002)*, pages 617–21, Stockholm, Sweden.

[Onori et al., 2008a] Onori, M., Semere, D., and Barata, J. (2008a). EUPASS Reference Architecture. *Deliverable of the EUPASS Project*.

[Onori et al., 2008b] Onori, M., Semere, D., and Barata, J. (2008b). Evolvable Assembly Systems: From Evaluation to Application. *International Conference on Information Technology for Balanced Automation Systems – Innovation in Manufacturing Networks*.

[OPC Foundation, 2006] OPC Foundation (2006). OPC Unified Architecture Specification - Part 5: Information Model .

[OPC Foundation, 2008] OPC Foundation (2008). OPC Unified Architecture (OPC-UA) Specifications. Available at http://www.opcfoundation.org/UA.

[Open Mobile Alliance, 2008] Open Mobile Alliance (2008). OMA Device Management protocol – v1.2. Open Mobile Alliance release.

[OpenWBEM, 2006] OpenWBEM (2006). OpenWBEM Project. Available at http://openwbem.sourceforge.net/.

[Ouksel and Sheth, 1999] Ouksel, A. and Sheth, A. (1999). Semantic interoperability in global information systems. *ACM Sigmod Record*, 28(1):5–12.

[Panetto and Molina, 2008] Panetto, H. and Molina, A. (2008). Enterprise integration and interoperability in manufacturing systems: Trends and issues. *Computers in industry*, 59(7):641–646.

[Paolucci et al., 2007] Paolucci, M., Wagner, M., and Martin, D. (2007). Grounding OWL-S in SAWSDL. In *Service-Oriented Computing*, pages 416–421. Springer.

[Papazoglou and Georgakopoulos, 2003] Papazoglou, M. and Georgakopoulos, D. (2003). Service-oriented computing. *Communications of the ACM*, 46(10):25–28.

[Papazoglou et al., 2007] Papazoglou, M., Traverso, P., Dustdar, S., and Leymann, F. (2007). Service-oriented computing: State of the art and research challenges. *Computer*, 40(11):38–45.

[Papazoglou and Van den Heuvel, 2005] Papazoglou, M. and Van den Heuvel, W. (2005). Web services management: A survey. *Internet Computing*, 9(6):58–64.

[Park and Ram, 2004] Park, J. and Ram, S. (2004). Information systems interoperability: What lies beneath? *ACM Transactions on Information Systems (TOIS)*, 22(4):595–632.

[Patrick, 2005] Patrick, P. (2005). Impact of SOA on enterprise information architectures. In *International Conference on Management of Data*, pages 844–848. ACM.

[Paul, 2005] Paul, R. (2005). DoD towards software services. In *International Workshop on Object-Oriented Real-Time Dependable Systems*, pages 3–6. IEEE.

[Pautasso et al., 2008] Pautasso, C., Zimmermann, O., and Leymann, F. (2008). Restful web services vs. big'web services: making the right architectural decision. In *International conference on World Wide Web*, pages 805–814. ACM.

[Pavlou et al., 2004] Pavlou, G., Flegkas, P., Gouveris, S., and Liotta, A. (2004). On management technologies and the potential of web services. *IEEE Communications Magazine*, 42(7):58–66.

[Pavon et al., 1998] Pavon, J., Tomas, J., Bardout, Y., and Hauw, L. (1998). CORBA for Network and Service Management in the TINA Framework. *IEEE Communications Magazine*, 36(3):72–79.

[Payne and Lassila, 2004] Payne, T. and Lassila, O. (2004). Semantic web services. *IEEE Intelligent Systems*, 19(4):14–15.

[Pěchouček and Mařík, 2008] Pěchouček, M. and Mařík, V. (2008). Industrial deployment of multi-agent technologies: review and selected case studies. *Autonomous Agents and Multi-Agent Systems*, 17(3):397–431.

[Peltz, 2003a] Peltz, C. (2003a). Web services orchestration: a review of emerging technologies, tools, and standards. Technical report, Hewlett-Packard.

[Peltz, 2003b] Peltz, C. (2003b). Web services orchestration and choreography. *Computer*, 36(10):46–52.

[Peltz and Anagol-Subbarao, 2004] Peltz, C. and Anagol-Subbarao, A. (2004). Design strategies for web services versioning. *Web Services Journal*.

[Perry and Denn, 2002] Perry, J. and Denn, R. (2002). *Java Management Extensions*. O'Reilly & Associates, Inc.

[Pine and Davis, 1999] Pine, B. and Davis, S. (1999). *Mass customization: The new frontier in business competition*. Harvard Business School Pr.

[PLCopen, 2009] PLCopen (2009). PLCopen TC6 – XML Schemes. Available at http://www.plcopen.org/.

[Pohl et al., 2008] Pohl, A., Krumm, H., Holland, F., Luck, I., and Stewing, F. (2008). Service-orientation and flexible service binding in distributed automation and control systems. In *International Conference on Advanced Information Networking and Applications-Workshops*, pages 1393–1398. IEEE.

[Potter and Whiteley, 2008] Potter, T. and Whiteley, B. (2008). PyWBEM – Python WBEM Client and Provider Interface. Available at http://pywbem.wiki.sourceforge.net/.

[Poulin, 2006] Poulin, M. (2006). Service versioning for SOA. *SOA World Magazine*, 26.

[Pras and Martin-Flatin, 2007] Pras, A. and Martin-Flatin, J. (2007). What can web services bring to integrated management? *Handbook of network and system administration*, page 241.

[PreManus, 2012] PreManus (2012). FP7-ICT PreManus project website. http://www.premanus.eu/.

[Puttonen et al., 2008] Puttonen, J., Lobov, A., and Martinez Lastra, J. (2008). An application of BPEL for service orchestration in an industrial environment. In *International Conference on Emerging Technologies and Factory Automation*, pages 530–537. IEEE.

[Ranky, 1986] Ranky, P. (1986). *Computer integrated manufacturing*. Prentice-Hall, Inc.

[Revelytix, 2012] Revelytix (2012). Knoodl - Distributed Information Management Systems (DIMS). Available at http://www.knoodl.com/.

[RI-MACS, 2012] RI-MACS (2012). FP7-ICT RI-MACS project website. http://www.rimacs.org.

[Ribeiro et al., 2010] Ribeiro, L., Barata, J., Cândido, G., and Onori, M. (2010). Evolvable Production Systems: An Integrated View on Recent Developments. In *CIRP-Sponsored International Conference on Digital Enterprise Technology*, pages 841–854. Springer.

[Ribeiro et al., 2008a] Ribeiro, L., Barata, J., and Colombo, A. (2008a). MAS and SOA: A Case Study Exploring Principles and Technologies to Support Self-Properties in Assembly Systems. In *International Conference on Self-Adaptive and Self-Organizing Systems*, pages 192–197. IEEE.

[Ribeiro et al., 2008b] Ribeiro, L., Barata, J., Colombo, A., and Jammes, F. (2008b). A generic communication interface for DPWS-based web services. In *International Conference on Industrial Informatics*, pages 762–767. IEEE.

[Ribeiro et al., 2008c] Ribeiro, L., Barata, J., and Mendes, P. (2008c). MAS and SOA: Complementary Automation Paradigms. *Innovation in Manufacturing Networks*, 266:259–268.

[Ribeiro et al., 2008d] Ribeiro, L., Barata, J., Onori, M., and Amado, A. (2008d). OWL ontology to support evolvable assembly systems. In *Intelligent Manufacturing Systems*, volume 9, pages 290–295.

[Ricci et al., 2007] Ricci, A., Buda, C., and Zaghini, N. (2007). An agent-oriented programming model for SOA & web services. In *International Conference on Industrial Informatics*, volume 2, pages 1059–1064. IEEE.

[Richardson and Ruby, 2007] Richardson, L. and Ruby, S. (2007). *RESTful web services*. O'Reilly Media.

[Roman et al., 2005] Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., and Fensel, D. (2005). Web Service Modeling Ontology. *Applied ontology*, 1(1):77–106.

[Rosen et al., 2008] Rosen, M., Lublinsky, B., Smith, K., and Balcer, M. (2008). *Applied SOA: Service-Oriented Architecture and Design Strategies*. Wiley India Pvt. Ltd.

[Rumbaugh et al., 2004] Rumbaugh, J., Jacobson, I., and Booch, G. (2004). *The Unified Modeling Language Reference Manual*. Pearson Higher Education.

[Samaras et al., 2013] Samaras, I. K., Hassapis, G. D., and Gialelis, J. V. (2013). A Modified DPWS Protocol Stack for 6LoWPAN-Based Wireless Sensor Networks. *IEEE Transactions on Industrial Informatics*.

[Sarimveis et al., 2008] Sarimveis, H., Patrinos, P., Tarantilis, C., and Kiranoudis, C. (2008). Dynamic modeling and control of supply chain systems: A review. *Computers and Operations Research*, 35(11):3530–3561.

[Sarjoughian et al., 2008] Sarjoughian, H., Kim, S., Ramaswamy, M., and Yau, S. (2008). A simulation framework for service-oriented computing systems. In *Winter Simulation Conference*, pages 845–853. IEEE.

[Sauter, 2005] Sauter, T. (2005). Integration aspects in automation – a technology survey. In *Conference on Emerging Technologies and Factory Automation*, volume 2, pages 9–pp. IEEE.

[Sauter, 2007] Sauter, T. (2007). The continuing evolution of integration in manufacturing automation. *Industrial Electronics Magazine*, 1(1):10–19.

[Schekkerman, 2003] Schekkerman, J. (2003). Enterprise Architecture Validation – Achieving Business-aligned and Validated Enterprise Architectures. Technical report, Institute For Enterprise Architecture Developments.

[Schepers et al., 2008] Schepers, T., Iacob, M., and Van Eck, P. (2008). A lifecycle approach to SOA governance. In *Symposium on Applied Computing*, pages 1055–1061. ACM.

[Schleipen, 2008] Schleipen, M. (2008). OPC UA supporting the automated engineering of production monitoring and control systems. In *Conference on Emerging Technologies and Factory Automation*, pages 640–647. IEEE.

[Schlenoff et al., 1998] Schlenoff, C., Ivester, R., and Knutilla, A. (1998). A robust process ontology for manufacturing systems integration. In *International Conference on Engineering Design and Automation*, pages 7–14.

[Schmelzer, 2006] Schmelzer, R. (2006). Solving the service granularity challenge. Available at http://www.zapthink.com/2006/03/09/solving-the-service-granularity-challenge/.

[Schmidt et al., 2005] Schmidt, M., Hutchison, B., Lambros, P., and Phippen, R. (2005). The enterprise service bus: making service-oriented architecture real. *IBM Systems Journal*, 44(4):781–797.

[Schneider and Kamiya, 2011] Schneider, J. and Kamiya, T. (2011). Efficient XML interchange (EXI) format 1.0 – W3C Working Draft. Available at http://www.w3.org/TR/exi/.

[Schoenwaelder, 2003] Schoenwaelder, J. (2003). Overview of the 2002 IAB network management workshop. Available at http://tools.ietf.org/html/rfc3535.

[Scholz et al., 2009] Scholz, A., Gaponova, I., Sommer, S., Kemper, A., Knoll, A., Buckl, C., Heuer, J., and Schmitt, A. (2009). eSOA-Service Oriented Architectures adapted for

embedded networks. In *International Conference on Industrial Informatics*, pages 599–605. IEEE.

[Schoop et al., 2002] Schoop, R., Colombo, A., Suessmann, B., and Neubert, R. (2002). Industrial experiences, trends and future requirements on agent-based intelligent automation. In *Annual Conference of the IEEE Industrial Electronics Society*, volume 4, pages 2978–2983. IEEE.

[Schulte and Natis, 1996] Schulte, R. and Natis, Y. (1996). Service oriented architectures, part 1. *Gartner, SSA Research Note SPA-401-068*.

[Sedukhin, 2005] Sedukhin, I. (2005). Web services distributed management: Management of web services (WSDM-MOWS) 1.0. OASIS-Standard.

[Seilonen et al., 2011] Seilonen, I., Tuomi, A., Olli, J., and Koskinen, K. (2011). Service-Oriented Application Integration for condition-based maintenance with OPC Unified Architecture. In *International Conference on Industrial Informatics*, pages 45–50. IEEE.

[Self-Learning, 2012] Self-Learning (2012). FP7-NMP Self-Learning Project website. http://selflearning.eu/.

[SemanticWeb, 2010] SemanticWeb (2010). SemanticWeb.org website. http://semanticweb.org.

[Sethi and Sethi, 1990] Sethi, A. and Sethi, S. (1990). Flexibility in manufacturing: a survey. *International Journal of Flexible Manufacturing Systems*, 2(4):289–328.

[Shadbolt et al., 2006] Shadbolt, N., Hall, W., and Berners-Lee, T. (2006). The semantic web revisited. *IEEE Intelligent Systems*, 21(3):96–101.

[Shah and Ward, 2003] Shah, R. and Ward, P. (2003). Lean manufacturing: context, practice bundles, and performance. *Journal of Operations Management*, 21(2):129–149.

[Shelby and Bormann, 2011] Shelby, Z. and Bormann, C. (2011). *6LoWPAN: The wireless embedded Internet*, volume 43. Wiley.

[Shen et al., 2007] Shen, W., Hao, Q., Wang, S., Li, Y., and Ghenniwa, H. (2007). An agent-based service-oriented integration architecture for collaborative intelligent manufacturing. *Robotics and Computer-Integrated Manufacturing*, 23(3):315–325.

[Shoch et al., 1982] Shoch, J., Dalal, Y., Redell, D., and Crane, R. (1982). Evolution of the Ethernet local computer network. *Computer*, 15(8):10–27.

[Sims, 2005] Sims, O. (2005). Developing the architectural framework for SOA – Part 2: service granularity and dependency management. *CBDI Forum Journal*.

[Singh and Huhns, 2005] Singh, M. and Huhns, M. (2005). *Service-oriented computing: semantics, processes, agents*. John Wiley & Sons Inc.

[SIRENA, 2005] SIRENA (2005). ITEA SIRENA Project website. http://www.sirena-itea.org.

[Sirin et al., 2007] Sirin, E., Parsia, B., Grau, B., Kalyanpur, A., and Katz, Y. (2007). Pellet: A practical OWL-DL reasoner. *Web Semantics: science, services and agents on the World Wide Web*, 5(2):51–53.

[Skadron et al., 2003] Skadron, K., Martonosi, M., August, D., Hill, M., Lilja, D., and Pai, V. (2003). Challenges in computer architecture evaluation. *Computer*, 36(8):30–36.

[Sleman and Moeller, 2008] Sleman, A. and Moeller, R. (2008). Integration of wireless sensor network services into other home and industrial networks; using device profile for web services (dpws). In *International Conference on Information and Communication Technologies: From Theory to Applications*, pages 1–5. IEEE.

[SOA4D, 2009a] SOA4D (2009a). SOA4D Forge - Service-Oriented Architecture for Devices. Available at https://forge.soa4d.org/.

[SOA4D, 2009b] SOA4D (2009b). WS-Management for DPWS. Available at https://forge.soa4d.org/projects/dpwsmanagement/.

[SOCRADES, 2007a] SOCRADES (2007a). Deliverable D1.2: Requirements of end users and component vendors/system integrators. Technical report, EU FP6 IST-5-034116 SOCRADES Consortium.

[SOCRADES, 2007b] SOCRADES (2007b). Deliverable D7.1: User requirements for the application systems engineering and lifecycle support of distributed smart embedded devices. Technical report, EU FP6 IST-5-034116 SOCRADES Consortium.

[SOCRADES, 2009] SOCRADES (2009). FP7-IST SOCRADES project website. http://www.socrades.eu/.

[SODA, 2009] SODA (2009). ITEA SODA Project website. http://www.soda-itea.org/.

[SofTulz, 2009] SofTulz (2009). Purgos. http://softulz.net/products.shtml.

[Son and Yi, 2012] Son, M. and Yi, M. (2012). OPC UA Discovery: A Study of Challenges and Perspective. *Applied Mechanics and Materials*, 157:110–113.

[Spiess et al., 2009] Spiess, P., Karnouskos, S., Guinard, D., Savio, D., Baecker, O., Souza, L., and Trifa, V. (2009). SOA-based Integration of the Internet of Things in Enterprise Services. In *International Conference on Web Services*, pages 968–975. IEEE.

[Spiess et al., 2008] Spiess, P., Nguyen, D., Weber, I., Markovic, I., and Beigl, M. (2008). Modelling, simulation, and performance analysis of business processes involving ubiquitous systems. In *Advanced Information Systems Engineering*, pages 579–582. Springer.

[Staab, 2009] Staab, S. (2009). *Handbook on ontologies*. Springer Verlag.

[Stake, 1995] Stake, R. (1995). *The art of case study research*. Sage Publications, Inc.

[Stallings, 1996] Stallings, W. (1996). IPv6: The new internet protocol. *IEEE Communications Magazine*, 34(7):96–108.

[Stallings, 1998] Stallings, W. (1998). SNMPv3: A security enhancement for SNMP. *IEEE Communications Surveys & Tutorials*, 1(1):2–17.

[Staudinger, 2012] Staudinger (2012). Staudinger Gmbh Simulation website. http://www.staudinger-est.de/en/simulation/.

[Stecke, 1985] Stecke, K. (1985). Design, planning, scheduling, and control problems of flexible manufacturing systems. *Annals of Operations Research*, 3(1):1–12.

[Sun Microsystems, 2004] Sun Microsystems (2004). WBEM Services – Java Web Based Enterprise Management. Available at http://wbemservices.sourceforge.net.

[SWSI, 2009] SWSI (2009). Semantic Web Services Initiative. http://www.swsi.org.

[ter Beek et al., 2007] ter Beek, M., Bucchiarone, A., and Gnesi, S. (2007). Web service composition approaches: From industrial standards to formal methods. In *International Conference on Internet and Web Applications and Services*, pages 15–15. IEEE.

[Terziyan and Katasonov, 2009] Terziyan, V. and Katasonov, A. (2009). Global understanding environment: Applying semantic and agent technologies to industrial automation. *Emerging Topics and Technologies in Information Systems, IGI Global*, pages 55–87.

[Terziyan and Kononenko, 2003] Terziyan, V. and Kononenko, O. (2003). Semantic Web enabled Web services: State-of-art and industrial challenges. *Lecture notes in computer science*, pages 183–197.

[Terziyan and Zharko, 2003] Terziyan, V. and Zharko, A. (2003). Semantic web and peer-to-peer: Integration and interoperability in industry. *International Journal of Computers, Systems and Signals*, 4(2):33.

[Tham and Buyya, 2005] Tham, C. and Buyya, R. (2005). Sensorgrid: Integrating sensor networks and grid computing. *CSI Communications*, 29(1):24–29.

[Thamboulidis et al., 2007] Thamboulidis, K., Koumoutsos, G., and Doukas, G. (2007). Semantic web services in the development of distributed control and automation systems. In *International Conference on Robotics and Automation*, pages 2940–2945. IEEE.

[The Open Group, 2008] The Open Group (2008). OpenPegasus – C++ CIM/WBEM Manageability Services Broker. http://www.openpegasus.org.

[The Open Group, 2012] The Open Group (2012). Open Management Infrastructure (OMI) – CIM/WBEM Manageability Services Broker. http://www.opengroup.org/software/omi.

[Theorin et al., 2012] Theorin, A., Ollinger, L., and Johnsson, C. (2012). Service-Oriented Process Control with Grafchart and the Devices Profile for Web Services. In *IFAC Symposium on Information Control Problems in Manufacturing, INCOM*.

[Thiagarajan et al., 2002] Thiagarajan, R., Srivastava, A., Pujari, A., and Bulusu, V. (2002). BPML: A process modeling language for dynamic business models. In *Advanced Issues of E-Commerce and Web-Based Information Systems*, pages 222–224. IEEE.

[Thomesse, 2005] Thomesse, J. (2005). Fieldbus technology in industrial automation. *Proceedings of the IEEE*, 93(6):1073–1101.

[Thompson, 1998] Thompson, J. (1998). Web-based enterprise management architecture. *Communications Magazine, IEEE*, 36(3):80–86.

[Thramboulidis et al., 2008] Thramboulidis, K., Doukas, G., and Koumoutsos, G. (2008). A SOA-based embedded systems development environment for industrial automation. *EURASIP Journal on Embedded Systems*, 2008:3.

[Tian et al., 2003] Tian, M., Gramm, A., Naumowicz, T., Ritter, H., and Freie, J. (2003). A concept for QoS integration in web services. In *International Conference on Web Information Systems Engineering*, pages 149–155. IEEE.

[TopQuadrant, 2012] TopQuadrant (2012). TopBraid Composer. Available at http://www.topquadrant.com.

[Tsai et al., 2006a] Tsai, W., Chen, Y., Bitter, G., and Miron, D. (2006a). Introduction to service-oriented computing. Technical report, Arizona State University.

[Tsai et al., 2006b] Tsai, W., Fan, C., Chen, Y., and Paul, R. (2006b). DDSOS: A dynamic distributed service-oriented simulation framework. In *Annual Symposium on Simulation*, pages 160–167. IEEE Computer Society.

[Tsai et al., 2006c] Tsai, W., Lee, Y., Cao, Z., Chen, Y., and Xiao, B. (2006c). RTSOA: Real-time service-oriented architecture. In *International Workshop on Service-Oriented System Engineering*, pages 49–56. IEEE.

[Tsarkov and Horrocks, 2006] Tsarkov, D. and Horrocks, I. (2006). FaCT++ description logic reasoner: System description. *Automated Reasoning*, pages 292–297.

[Ueda, 1992] Ueda, K. (1992). A concept for bionic manufacturing systems based on DNA-type information. In *International PROLAMAT Conference on Human Aspects in Computer Integrated Manufacturing*, pages 853–863. North-Holland Publishing Co.

[Uschold and Gruninger, 1996] Uschold, M. and Gruninger, M. (1996). Ontologies: Principles, methods and applications. *Knowledge engineering review*, 11(2):93–136.

[Valckenaers et al., 1997] Valckenaers, P., Van Brussel, H., Bongaerts, L., and Wyns, J. (1997). Holonic manufacturing systems. *Integrated Computer-Aided Engineering*, 4(3):191–201.

[Van Brussel et al., 1998] Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L., and Peeters, P. (1998). Reference architecture for holonic manufacturing systems: PROSA. *Computers in industry*, 37(3):255–274.

[Vernadat, 2007] Vernadat, F. (2007). Interoperable enterprise systems: principles, concepts, and methods. *Annual Reviews in Control*, 31(1):137–145.

[Villaseñor Herrera et al., 2011] Villaseñor Herrera, V., Vidales Ramos, A., and Martínez Lastra, J. (2011). An agent-based system for orchestration support of web service-enabled devices in discrete manufacturing systems. *Journal of Intelligent Manufacturing*, pages 1–22.

[Vinoski, 2003] Vinoski, S. (2003). Integration with web services. *Internet Computing, IEEE*, 7(6):75–77.

[Vyatkin, 2007] Vyatkin, V. (2007). *IEC 61499 function blocks for embedded and distributed control systems design*. ISA-Instrumentation, Systems, and Automation Society.

[W3C, 2002] W3C (2002). Web Services Activity. http://www.w3.org/2002/ws/.

[W3C, 2005] W3C (2005). Web services choreography description language. Web Services Choreography Working Group – Available at http://www.w3.org/TR/ws-cdl-10/.

[W3C, 2008] W3C (2008). Semantic Web Services Interest Group. http://www.w3.org/2002/ws/swsig/.

[W3C, 2010] W3C (2010). W3C Semantic Web Activity website. http://www.w3.org/2001/sw/.

[W3C, 2012] W3C (2012). World wide web consortium. http://www.w3.org/.

[Wang et al., 2005] Wang, Z., Xu, X., and Zhan, D. (2005). A survey of business component identification methods and related techniques. *International Journal of Information Technology*, 2(4):229–238.

[Warrier and Besaw, 1989] Warrier, U. and Besaw, L. (1989). Common Management Information Services and Protocol over TCP/IP (CMOT). Available at http://www.hjp.at/doc/rfc/rfc1095.html.

[Weaver, 2001] Weaver, A. C. (2001). Survey of industrial information technology. In *Annual Conference of the IEEE Industrial Electronics Society*, volume 3, pages 2056–2061. IEEE.

[Westerinen and Bumpus, 2003] Westerinen, A. and Bumpus, W. (2003). The continuing evolution of distributed systems management. *IEICE Transactions on Information and Systems*, 86(11):2256–2261.

[White, 1987] White, L. (1987). Software testing and verification. *Advances in Computers*, 26(1):335–390.

[White, 2004] White, S. (2004). Introduction to BPMN. *IBM Cooperation*, pages 2008–2029.

[Wieczorek et al., 2008] Wieczorek, S., Roth, A., Stefanescu, A., and Charfi, A. (2008). Precise steps for choreography modeling for SOA validation and verification. In *International Symposium on Service-Oriented System Engineering*, pages 148–153. IEEE.

[Wilkes and Veryard, 2004] Wilkes, L. and Veryard, R. (2004). Service-oriented architecture: Considerations for agile systems. *Microsoft Architects Journal*, 2:11–23.

[Wiseman Group, 2008] Wiseman Group (2008). Wiseman - A Java implementation of WS-Management. Available at https://wiseman.dev.java.net/.

[Woods and Mattern, 2006] Woods, D. and Mattern, T. (2006). *Enterprise SOA: designing IT for business innovation*. O'Reilly Media.

[Wooldridge, 2002] Wooldridge, M. (2002). An introduction to multiagent systems. *West Sussex, England: John Wiley and Sons Ltd*, 348.

[Wooldridge and Jennings, 1998] Wooldridge, M. and Jennings, N. (1998). Pitfalls of agent-oriented development. In *Second International Conference on Autonomous Agents*, pages 385–391.

[WS4D, 2012a] WS4D (2012a). Web Services for Devices – DPWS Explorer v3.4. Available at http://ws4d.e-technik.uni-rostock.de/dpws-explorer/.

[WS4D, 2012b] WS4D (2012b). Web Services for Devices – WS4D-JMEDS stack. Available at http://ws4d.e-technik.uni-rostock.de/jmeds/.

[Yemini, 1993] Yemini, Y. (1993). The OSI network management model. *Communications Magazine, IEEE*, 31(5):20–29.

[Yeung, 2007] Yeung, W. (2007). CSP-based verification for Web service Orchestration and Choreography. *Simulation*, 83(1):65–74.

[Zeeb et al., 2007a] Zeeb, E., Bobek, A., Bohn, H., and Golatowski, F. (2007a). Service-oriented architectures for embedded systems using devices profile for web services. In

International Conference on Advanced Information Networking and Applications, volume 1, pages 956–963. IEEE.

[Zeeb et al., 2007b] Zeeb, E., Bobek, A., Bohn, H., Pruter, S., Pohl, A., Krumm, H., Luck, I., Golatowski, F., and Timmermann, D. (2007b). WS4D: SOA-Toolkits making embedded systems ready for Web Services. *Open Source Software and Product lines (OSSPL07)*, 10(1.73):1088.

[Zelkowitz, 2009] Zelkowitz, M. (2009). An update to experimental models for validating computer technology. *Journal of Systems and Software*, 82(3):373–376.

[Zelkowitz and Wallace, 1998] Zelkowitz, M. and Wallace, D. (1998). Experimental models for validating technology. *Computer*, 31(5):23–31.

[Zhao et al., 2010] Zhao, F., Chen, J., Dong, G., and Guo, L. (2010). SOA-based remote condition monitoring and fault diagnosis system. *The International Journal of Advanced Manufacturing Technology*, 46(9):1191–1200.

[Zur Muehlen et al., 2005] Zur Muehlen, M., Nickerson, J., and Swenson, K. (2005). Developing web services choreography standards – the case of REST vs. SOAP. *Decision Support Systems*, 40(1):9–29.